NOVA IMS Information Management School

DSAA

Mestrado em Data Science and Advanced Analytics

Master Program in Data Science and Advanced Analytics

An Artificial Intelligence Method to Describe the Onset and Transition from Stochastic to Coordinated Neural Activity in Danionella Translucida Embryo

Inês Filipa Ferreira Diogo

Dissertation presented as partial requirement for obtaining the Master's degree in Data Science and Advanced Analytics

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

AN ARTIFICIAL INTELLIGENCE METHOD TO DESCRIBE THE ONSET AND TRANSITION FROM STOCHASTIC TO COORDINATED NEURAL ACTIVITY IN DANIONELLA TRANSLUCIDA EMBRYO

by

Inês Filipa Ferreira Diogo

Dissertation presented as partial requirement for obtaining the Master's degree in Data Science and Advanced Analytics

Advisor: Mauro Castelli

Co-Advisor*:* Davide Accardi

November 2021

ACKNOWLEDGEMENTS

I would like to thank my advisors, Professor Mauro Castelli and Dr. Davide Accardi, and also Dr. Michael Orger for envisioning the project and giving me the opportunity to carry on with it. Professor Mauro Castelli for always being available to give me advice and guidance. Dr. Davide Accardi for the guidance and for allowing me to be part of the ABBE Platform (Advanced BioImaging & BioOptics Experimental Platform) of the Champalimaud Foundation throughout this project. Michael Orger for the invaluable support related to the biological field. Thank you very much.

I would like to thank Leonor Morgado not only because she was co-responsible to carry on this project, being focused on sample handling and image acquisition, but also for showing me and explaining the entire process and for all the help and advice. Thank you.

I would also like to thank Anna Pezzarossa. Thank you very much for all your help and advice, for both the image analysis and the writing process, and also for the encouragements.

To my parents for always being by my side, always believing in me, and always wanting the best for me. To Flapas for always making me smile and being part of my life. Thank you so much for all the love and support.

To my friends for being there for me and all the encouragement, especially one of my best friends, Inês Ribeiro. Thank you for everything.

To my uncles and cousins who were also there for me. Thank you.

Finally, a special thank you to everyone who had a massive positive impact on my life during this phase, even those who unknowingly helped me a lot.

ABSTRACT

In recent years, deep learning has become increasingly successful applied to tackle different questions in several fields. In bioimage analysis, it has been used to extract meaningful information from microscopy images. Here, we applied DL to light-sheet microscopy data to understand the early development of the nervous system. It is currently known that the brain is responsible for most of our voluntary and involuntary actions, and that it regulates physiological processes throughout the body. However, technical barriers have left us with many open questions regarding the development and function of neural circuits. Imaging has proven a powerful technique to answer these questions, although difficulties in segmenting and tracking single neurons have slowed down progress.

Danionella translucida has recently been introduced as a powerful model organism for neuroscience studies due to having the smallest known vertebrate brain and not developing a complete skull as an adult, thus making it easily accessible for imaging studies. Yet, the emergence of neural activity and subsequent assembly of neural circuits in the early embryo development has not been characterized.

This dissertation intends to provide a first description of the entire process at cellular resolution by using advanced microscopy techniques and an artificial intelligence method to segment and analyze the data. We use light-sheet fluorescence microscopy to image the onset and coordination of Danionella translucida spinal cord neural activity with high temporal resolution and for long periods of time. Moreover, we analyze the data with a deep learning-based algorithm to detect, segment and track in space and time the signal of each neuron.

We focused our analysis on the intensity peaks of the signal, in other words, the moment the neurons were firing, and we found more activity in the posterior region of the embryo, suggesting a correspondence to the extension of the tail.

This work demonstrates that the combination of methods used was able to image and analyze the data successfully. It opens the possibilities for a more in-depth study of the Danionella translucida neural network, and for studying signals from crowded images with single-cell resolution that otherwise would be too complex to be analyzed.

KEYWORDS

Deep Learning; Cell Detection; Cell Segmentation; Light-sheet Fluorescence Microscopy; Danionella Translucida; Neural Activity

RESUMO

Nos últimos anos, a aprendizagem profunda tem se tornado cada vez mais bem-sucedida quando aplicada para lidar com diferentes questões em diversos campos. Na análise de bioimagem, tem sido usada para extrair informações significativas de imagens microscópicas, onde aplicamos aprendizagem profunda a dados de microscopia de *light-sheet* para compreender o desenvolvimento inicial do sistema nervoso. Atualmente, sabe-se que o cérebro é responsável pela maioria de nossas ações voluntárias e involuntárias e que regula os processos fisiológicos em todo o corpo. No entanto, as barreiras técnicas deixaram muitas questões em aberto em relação ao desenvolvimento e função dos circuitos neuronais. Imagiologia provou ser uma técnica poderosa para responder a essas perguntas, embora as dificuldades em segmentar e rastrear neurônios individuais tenham retardado o progresso.

Danionella translucida foi recentemente introduzida como um poderoso organismo modelo para estudos neurocientíficos devido a ter o menor cérebro de vertebrado conhecido e não desenvolver um crânio completo na idade adulta, tornando-a facilmente acessível para estudos de imagem. No entanto, o surgimento da atividade neural e subsequente montagem de circuitos neurais no desenvolvimento inicial do embrião não foi ainda caracterizado.

Esta dissertação pretende fornecer uma descrição inicial de todo o processo de resolução celular, utilizando técnicas avançadas de microscopia e um método de inteligência artificial para segmentar e analisar os dados. Usamos microscopia de fluorescência de *light-sheet* para obter imagens do início e da coordenação da atividade neuronal da medula espinhal da *Danionella translucida* com alta resolução temporal e por longos períodos de tempo. Além disso, analisamos os dados com um algoritmo baseado em aprendizagem profunda para detetar, segmentar e rastrear no espaço e no tempo o sinal de cada neurônio.

Focamos nossa análise nos picos de intensidade do sinal, ou seja, no momento em que os neurónios estavam a disparar, e encontramos mais atividade na região inferior do embrião, sugerindo uma correspondência com a extensão da cauda.

Este trabalho demonstra que a combinação de métodos utilizados foi capaz de gerar imagens e analisar os dados com sucesso. Abre as possibilidades para um estudo mais aprofundado da rede neuronal da *Danionella translucida*, e para estudar sinais de imagens aglomeradas com resolução de célula única que, de outra forma, seriam muito complexas para serem analisadas.

PALAVRAS-CHAVES

Aprendizagem profunda; Deteção de células; Segmentação de células; Microscopia de fluorescência de light-sheet; Danionella Translucida; Atividade Neuronal

INDEX

1. Introduction 1
   1.1. Research Question And Dissertation Objective 1
   1.2. Danionella Translucida 2
   1.3. Light-Sheet Fluorescence Microscopy 2
2. Theoretical Background 4
   2.1. Machine Learning 4
      2.1.1. Supervised Learning 4
      2.1.2. Unsupervised Learning 4
      2.1.3. Semi-supervised Learning 5
      2.1.4. Reinforcement Learning 5
   2.2. Artificial Neural Network 5
      2.2.1. Regularization 6
   2.3. Deep Learning 6
      2.3.1. Stochastic Gradient Descent 6
      2.3.2. Momentum 7
   2.4. Convolutional Neural Network 8
      2.4.1. Architecture of a CNN 8
         2.4.1.1. Input Layer 8
         2.4.1.2. Convolutional Layer 9
         2.4.1.3. Pooling Layer 9
         2.4.1.4. Fully Connected Layer 10
      2.4.2. Transfer Learning 10
   2.5. R-CNN 10
   2.6. Fast R-CNN 11
   2.7. Faster R-CNN 12
      2.7.1. Intersection over Union 13
      2.7.2. Non-Maximum Suppression 14
      2.7.3. Balanced Samples 14
      2.7.4. RoI Pooling Layer and Final Layer 15
   2.8. Mask R-CNN 15
      2.8.1. Instance Segmentation 16
      2.8.2. Mask Representation 16

2.8.3. RoI Align 16
2.8.4. Loss Function 17
2.9. U-Net 18
3. Literature review 20
3.1. StarDist 20
3.2. TrackMate 22
3.3. StarDist and TrackMate 22
3.4. Other applications 22
4. Data 24
4.1. Danionella Translucida Care 24
4.2. Data Acquisition 24
4.2.1. Sample Preparation 24
4.2.2. Light-sheet Imaging 25
4.3. Data Description 27
5. Methodology 30
5.1. Tools 30
5.2. Segmentation using StarDist 30
5.2.1. Validation 31
5.2.1.1. Comparison between StarDist and human segmentation 31
5.2.1.2. Comparison between StarDist and human segmentation performed on synthetic data 34
5.2.1.3. Conclusion on the validation 35
5.3. Intensities 35
5.4. Tracking File 36
5.5. Tracking using TrackMate 37
5.6. Table with Spots Information and Intensities 39
5.7. Distinguish Between Left and Right Side of the Embryo 40
5.8. Analysis of the Tables 42
6. Results 45
6.1. Neural activity of the embryos from the selected movies 45
6.1.1. Frequency of the Intensity peaks 45
6.1.2. Position of the intensity peaks over time 47
6.2. Neural activity at a later stage of embryonic development 49
6.2.1. Intensity over time of coordinated neurons 49
7. Discussion 52

7.1. Neural activity of the embryos from the selected movies 52
7.1.1. Frequency of the Intensity peaks 52
7.1.2. Intensity peaks' position in y over time 53
7.2. Neural activity at a later stage of embryonic development 53
7.2.1. Intensity over time of coordinated neurons 53
7.3. Performance of the software 54
8. Conclusions 55
8.1. Implications based on this project 55
8.2. Limitations 55
8.3. Future Work 56
9. Bibliography 58

LIST OF FIGURES

Figure 1.1: *Danionella Translucida* 2
Figure 1.2: Light-sheet Illumination 3
Figure 2.1: Momentum 7
Figure 2.2: Example of a simple CNN's architecture 8
Figure 2.3: Illustration of a convolutional layer 9
Figure 2.4: Architecture of R-CNN 11
Figure 2.5: Architecture of Fast R-CNN 12
Figure 2.6: Architecture of Faster R-CNN 12
Figure 2.7: Region Proposal Network process (RPN) 13
Figure 2.8: Intersection over Union (IoU) 13
Figure 2.9: Before and after applying Non-Maximum Suppression 14
Figure 2.10: Mask R-CNN architecture 15
Figure 2.11: RoIAlign 16
Figure 2.12: U-Net segmentation mask 18
Figure 2.13: U-Net Architecture 19
Figure 3.1: The number of scientific publications per year in the last decade (until 2019) on deep learning and related terms 20
Figure 3.2: StarDist 21
Figure 4.1: Illustrative image of the dechorionation procedure 25
Figure 4.2: Scheme for the preparation for light-sheet imaging 26
Figure 4.3: YZ orthogonal projection of an embryo 26
Figure 4.4: Representation of the acquired channels 27
Figure 4.5: Intensity during the time of the experiment 28
Figure 4.6: Representation of the intensity fading over time in a *Danionella translucida* embryo 28
Figure 4.7: Cropped frame from a *Danionella Translucida* embryo 29
Figure 5.1: StarDist segmentation 31
Figure 5.2: Example of the segmentation performed by a human and by StarDist 32
Figure 5.3: Distribution graph for the minimum distance between two centroids 33
Figure 5.4: Synthetic images 34
Figure 5.5: Graph comparing the number of segmented objects between StarDist, Experts, and Non-experts on the synthetic images 34
Figure 5.6: Graph comparing the number of segmented objects between StarDist and humans on the synthetic images 35

Figure 5.7: Comparison between the centroid from StarDist and the actual geometric center 36
Figure 5.8: Tracking file code 37
Figure 5.9: Centroids from the tracking file overlapping the original image.....37
Figure 5.10: Tracks in TrackMate 38
Figure 5.11: Skeleton of an embryo 41
Figure 5.12: Organized graph - Intensity over time 43
Figure 5.13: Graph with intensity peaks 44
Figure 6.1: Peaks Frequency.....46
Figure 6.2: Quantity of objects in time 47
Figure 6.3: Intensity peaks' position in y over time 48
Figure 6.4: Intensity in time of coordinated neurons 50
Figure 6.5: Sequence of frames showing the neurons firing 50
Figure 6.6: Intensity in time of uncoordinated neurons 51

LIST OF TABLES

Table 5.1: Intensity table.....36
Table 5.2: TrackMate table – Statistics about spots in tracks.....39
Table 5.3: TrackMate table with mean intensity for each object.....40
Table 5.4: TrackMate table with mean intensities and side information.....41
Table 5.5: Table with side information for each neuron 42
Table 5.6: Binary table – Existence of peaks.....44

LIST OF ABBREVIATIONS AND ACRONYMS

AI	Artificial Intelligence
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
DL	Deep Learning
DNN	Deep Neural Network
hpf	hour(s) post-fertilization
IoU	Intersection over Union
MIP	Maximum Intensity Projection
NMS	Non-Maximum Suppression
R-CNN	Region-based Convolutional Neuronal Network
ReLU	Rectified Linear Unit
RGB	Red-Green-Blue
RoI	Region of Interest
RPN	Region Proposal Network
SGD	Stochastic Gradient Descent
SVM	Support Vector Machine

1. INTRODUCTION

Over the past few years, deep learning has become progressively more used in several research areas due to its success in solving problems. A class of machine learning algorithms that can find out patterns and structures in high-dimensional data, thus able to handle an increasing amount of data, beating other techniques (Lecun et al., 2015). It has been utilized in recognition of speech and image, natural language understanding, and scientific and medical research (Goodfellow et al., 2016). The medical field can generate a large amount of data, especially when dealing with *in vivo* imaging during many hours. Consequently, automated algorithms for imaging analysis, such as detection, segmentation, and quantification of cells, are fundamental to making a more accurate and more straightforward diagnosis and disease research. Apart from allowing better analysis and comprehension of such a high quantity of data, manually detecting cells would be time-consuming, laborious, and highly susceptible to human error caused by some factors like cell concentration and visibility. Furthermore, human perception can lead to biased results and, consequently, variability between interobserver and intraobserver (Salinas et al., 1997).

For all the mentioned above, counting cells is an essential task for neuroscience, which is the scientific study of how the nervous system develops, how it is structured, and what it does. While it is known that the brain is responsible for most of our voluntary and involuntary actions, and it regulates physiological processes through the body, there are still many open questions regarding the development and function of neural circuits.

Nevertheless, there are limitations related to imaging the human nervous system. So, model organisms with similar characteristics to humans and more accessible to imaging, like *Danionella translucida*, can be used to study the nervous system.

1.1. RESEARCH QUESTION AND DISSERTATION OBJECTIVE

This dissertation aims to provide a first answer to the research question that is the unknown description of the onset and transition from stochastic to coordinated neural activity of *Danionella translucida* embryo.

We intend to answer this research question by using advanced microscopy techniques to capture the data. In particular, we used light-sheet microscopy to image *Danionella translucida* spinal cord with high temporal resolution and for long periods. Moreover, data analysis was performed by developing a deep learning-based algorithm that can detect, segment and track in space and time the signal of each neuron. Thus, providing a description of the entire process at cellular resolution.

1.2. DANIONELLA TRANSLUCIDA

Until now, in the neuroscientific field, one of the most used model organisms is zebrafish. However, *Danionella translucida* has recently been introduced due to its benefits over zebrafish.

Conventionally, zebrafish is used for neural studies since it is a well-established model organism because it shares a large proportion of the human genome and, consequently, shares many features with the human systems (Howe et al., 2013). Additionally, zebrafish embryos have a small size and a nearly optically transparent appearance, making them well suited for microscopy and allowing for more straightforward observation of the development of its internal structures (Hill et al., 2005). Another main reason for its usage is that the zebrafish grows at a high-speed rate and breeds readily (Kimmel et al., 1995). Zebrafish embryo has already been used to study neuronal circuit development (Wan et al., 2019).

Danionella translucida has been presented as a powerful model organism for neuroscience studies, given that it has the smallest known vertebrate brain and does not develop a complete skull as an adult. Besides, it has more than 85% of zebrafish genes, having the same advantages of being small and optical clearer, but it is also behaviorally and neural complex as an adult vertebrate (Schulze et al., 2018). Figure 1.1 shows the *Danionella translucida* size and transparency.

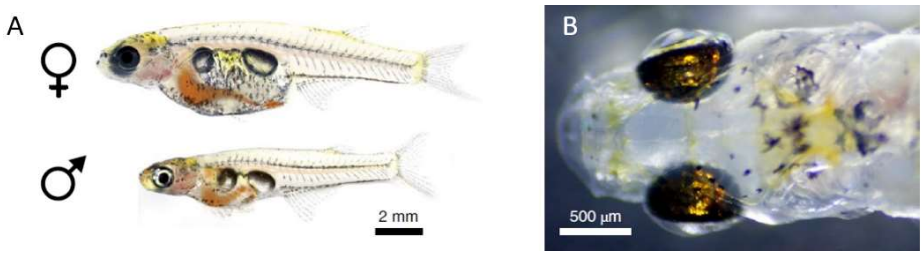


Figure 1.1: *Danionella Translucida*

A: body size of a female and male specimen, top and bottom respectively; B: Head image. Image from (Schulze et al., 2018).

Hence, it is a great promising model to study neural development. However, the emergence of the neural activity and subsequent assembly of neural circuits in the early embryo of *Danionella translucida* development has not yet been characterized.

1.3. LIGHT-SHEET FLUORESCENCE MICROSCOPY

Fluorescence microscopy uses fluorescence to examine cells, which is the process of illuminating a specimen with a light of a specific wavelength, with the purpose of the molecules absorbing it, causing them to emit light of longer wavelengths. This difference allows us only to visualize the

fluorescent cells, which are the ones that emit fluorescence. These molecules that can re-emit light upon excitation are called fluorophores (Lichtman & Conchello, 2005).

Light-sheet fluorescence microscopy has recently become widely used to image *in vivo* organisms over a long period of time since it combines gentle illumination with a high spatial-temporal resolution. In addition, it allows an easier rotation of the sample, which offers the possibility to acquire images from different angles (Weber & Huisken, 2011). In contrast to other imaging techniques, it manages to selectively illuminate only a thin slice of the sample, as shown in Figure 1.2, which means the fluorophores out of the focus plane are not excited, reducing photodamage (Reynaud et al., 2010).

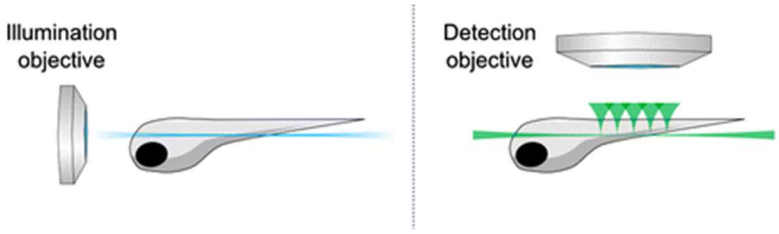


Figure 1.2: Light-sheet Illumination

The blue and green indicate the volume that is excited when imaging a single plane of the sample. Image adapted from (Wolf & Debrégeas, 2018)

Photodamage happens when the excited molecules become more reactive, making them more prone to react with other molecules, producing responses that can lead to potentially cell damage, resulting in photobleaching and phototoxicity (Satsoura et al., 2007). Photobleaching occurs when a molecule loses its ability to fluoresce, whereas phototoxicity is the damage of subcellular components and can even lead to cell and then sample death (Icha et al., 2017).

Therefore, the light-sheet technique enables us to image *Danionella translucida* embryos for more extended periods with high frequency and more efficiently.

2. THEORETICAL BACKGROUND

2.1. MACHINE LEARNING

Machine learning is a branch of artificial intelligence that focuses on developing computer algorithms able to automatically learn and improve through experience and by using data (Mitchell, 1997). The goal is to learn and recognize patterns on sample data, known as training data, in order to make predictions and decisions in the future without human intervention (Zhang, 2020). Throughout recent years, there has been an exponential increase in generated data. As a result, machine learning has been evolving and becoming progressively more used to analyze data thanks to its computational power. Consequently, it enables us to analyze an enormous amount of data while providing accurate and fast results.

There are various applications where this type of algorithm can be used, such as healthcare, sentiment analysis, fraud recognition, computer vision, and several other fields. Moreover, machine learning can be classified, considering the type of input data, into four general areas: Supervised Learning, Unsupervised Learning, Semi-Supervised, and Reinforcement Learning.

2.1.1. Supervised Learning

Supervised learning uses labeled data to train the model to predict or classify accurately. Given input data already categorized, usually by humans, the algorithm learns the features of the different labels by adjusting their weights until the model fits the dataset. After training, it can predict the output for new unseen data based on its features (Mohri et al., 2012).

Furthermore, supervised learning algorithms can be divided into classification and regression. In classification algorithms, the output value is categorical, so it is limited. Whereas, in regression algorithms, the outputs can have numerical values between a range. For example, a regression problem would predict a person's age, while a categorical problem would classify between young and old.

2.1.2. Unsupervised Learning

In supervised learning, the opposite occurs, the input dataset is unlabeled, and the algorithm's goal is to find patterns in the data (James et al., 2013). The system identifies shared characteristics in the data and gives an output based on the presence or absence of such traits. For instance, it can be used to group customers and understand their differences.

2.1.3. Semi-supervised Learning

In semi-supervised algorithms, the input data is both labeled and unlabeled, where generally, there is more unlabeled data than labeled. This way, it uses the labeled data to classify and the unlabeled to extract new features, significantly improving the model's accuracy.

2.1.4. Reinforcement Learning

Reinforcement learning algorithms learn from feedback by interacting with the environment and using trial and error. The learning system tries to give outputs and receives positive or negative feedback for each output. According to the feedback received, it is able to learn by itself and improve the performance of the task that is being learned (Zhang, 2020).

2.2. ARTIFICIAL NEURAL NETWORK

Artificial neural network (ANN) is inspired by the human brain, and, like the biological brain, it is composed of connected networks of neurons. Hence, ANNs are organized into multiple layers, where each one is responsible for processing different parts of the information. It starts with the input layer, which receives the data, followed by one or more hidden layers that process the information, and finally, an output layer generates the results.

The most simple neural network is the perceptron (Goodfellow et al., 2016). Furthermore, the single-layer perceptron, consisting of the input and output layers and just one hidden layer, was the first artificial network. However, this network is only capable of dealing with linearly separable patterns. Thus, to solve this problem, the multilayer perceptron was developed by combining multiple neurons to the network, where a neuron is a mathematical function.

The perceptron consists of an algorithm for supervised learning that produces an output based on a function that receives multiple inputs and combines weights and biases (Haykin et al., 2009). First, concerning the input data, the weights are defined in the training process and combined with the respective inputs through a summation function. Then, an activation function is used in every neuron to reduce the amplitude of the output. Finally, by using a loss function, the error is calculated with the difference between the output value and the ground truth, and it is propagated backward, updating the weights and the biases, based on a learning rate, which is a parameter that controls how much the error affects the values for the weights. This process is called backpropagation and is repeated until the error is minimized.

As a result, the computational power of neural networks is associated with their parallel distributed structure and their ability to generalize, that is, their ability to produce accurate results in unseen data (Haykin et al., 2009).

2.2.1. Regularization

One of the most common problems when dealing with neural networks is overfitting, which occurs when the algorithm performs very well on the training data but fails with unseen data of similar nature. A way to prevent this is L1 and L2 regularizations. These are techniques that reduce the complexity of the model during training by updating the cost function adding a regularization term to the loss, defined in equation (2.1).

$$Cost\ function = Loss\ function + term \tag{2.1}$$

L1 regularization adds a penalty parameter using the sum of the absolute value of the weights. In contrast, L2 regularization, also known as weight decay or Ridge regression, uses the squared value of weights (Goodfellow et al., 2016). L2 is the most commonly used since the weight parameters decrease but do not become zero, contrary to L1. Furthermore, it causes the learning algorithm to suppress any irrelevant components of the weights in the training process, leading to more accurate models (Krogh & Hertz, 1991).

2.3. DEEP LEARNING

Deep learning is a subfield of machine learning based on artificial neural networks. Machine learning algorithms learn from the input dataset, whereas deep learning algorithms can train themselves to detect or classify patterns in the raw data (Lecun et al., 2015). Consequently, it can be used in many fields, including computer vision, natural language processing, and medical image analysis. A deep neural network is an ANN with multiple hidden layers, which allows dealing with complex problems, since the system learns as it processes information through each hidden layer, just like the human brain.

In summary, compared with machine learning, it eliminates some pre-processing on the data since these algorithms can receive unstructured input data and can automate feature extraction. In addition, it uses processes of gradient descent and backpropagation to improve accuracy.

2.3.1. Stochastic Gradient Descent

Gradient descent is an optimization technique in Deep Learning and Machine Learning (Mitchell, 1997). The gradient is the slope of a function, so gradient descent is used to find a local minimum of a loss function by iteratively finding the optimal values of the parameters (i.e., the neural network's weights). The loss function measures the error between the prediction and the target values. Subsequently, the optimizer updates the network's weights by considering the value of the loss function. The magnitude of the update depends on the value of the learning rate.

A stochastic process has a random probability distribution. Therefore, in Stochastic Gradient Descent (SGD), only a subset, also called minibatch, is randomly selected to update a parameter in a particular iteration instead of the entire training sample. As a result, SGD reduces the computational complexity, performing faster iterations.

2.3.2. Momentum

Momentum is an optimizer used to accelerate the learning process. It accumulates an exponentially decaying moving average of the past gradients and moves in their direction (Goodfellow et al., 2016). In SGD, the derivate of the loss function is an estimation based on a minibatch, which means it is not always headed in the optimal direction since the derivates are noisy. Thus, using momentum with SGD is better because using exponentially weighted averages helps provide an estimation closer to the actual value of the derivate.

Another reason that it is better to use SGD with momentum lies in ravines. Ravine is an area where there is a steeper curve in one dimension than in another. Additionally, ravines are typically near local minimums, which makes SGD oscillate across the narrow ravine since the negative gradient will point out towards the sides rather than the optimum. Thus, momentum helps to accelerate the gradient descent in the optimal direction. Figure 2.1 illustrates this momentum effect, where the black arrows show the gradient's path without momentum, and the red path indicates the path followed by the gradient with momentum. This figure shows that the path with momentum is smaller than the one without momentum.

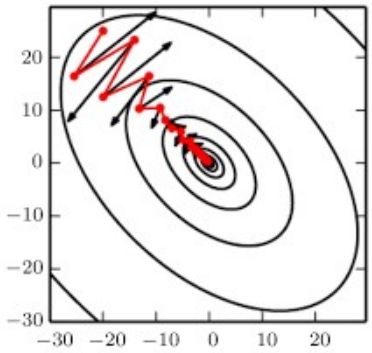


Figure 2.1: Momentum

Illustration of the Momentum Effect. The black contour lines represent a quadratic loss function. The red path indicates the path followed by the gradient with momentum, and the arrows indicate the direction that the gradient descent would take at that point (Goodfellow et al., 2016).

The momentum optimizer is defined by a parameter $\alpha \in [0.1[$, which is usually initialized with a small value and is slowly increased to 0.9 and closer to it.

2.4. CONVOLUTIONAL NEURAL NETWORK

Convolutional Neural Networks (CNN) are a class of ANNs commonly used in computer vision since it is specifically designed to process 2D grids of pixel values (Goodfellow et al., 2016).

In regular neural networks, the inputs are only processed in the forward direction. All the neurons in one layer pass information in the direction of the subsequent layer, so a single output node interacts with all the input nodes. This process would become computational complex when dealing with image data. Whereas CNNs use convolutional, which is a mathematical operation that processes information in a grid-like structure. Thus, making CNNs very successful in capturing spatial and temporal dependencies and performing image and video processing.

2.4.1. Architecture of a CNN

The architecture of a CNN was inspired by the biological processes of the virtual cortex path (Lecun et al., 2015). It contains an input layer, an output layer, and several hidden layers in between, as shown in figure 2.2. The hidden layers are composed first by convolutional and pooling layers, followed by fully connected layers.

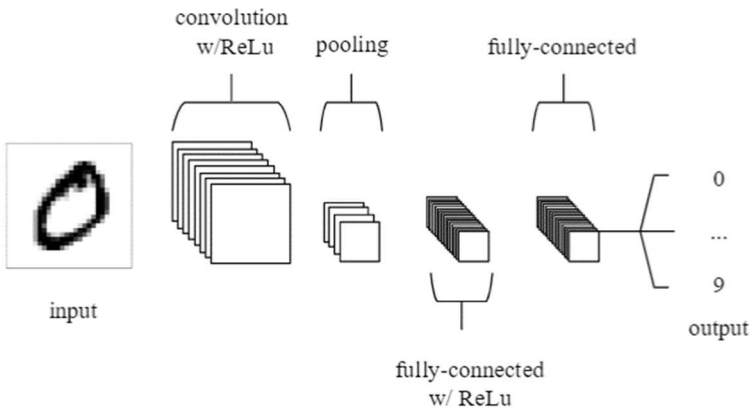


Figure 2.2: Example of a simple CNN's architecture

In this example, there are five layers. From left to right: The input image, the convolution layer with the activation function ReLU, the pooling layer, the fully connected layer with ReLU, another fully connected layer, and the final output layer. In this example, the goal was to identify which number was present in the input image. (O'Shea & Nash, 2015).

2.4.1.1. Input Layer

The input layer is the image that is going to be analyzed, with dimensions [width × height × depth], where depth is the number of colored channels in the image, there could be three channels in case

of a colored image (RGB) or one, in case of a grayscale image. Besides, an image can be defined as an array of values (pixels) which range from 0 to 255.

2.4.1.2. Convolutional Layer

The convolutional layer, as the name implies, plays an essential task in this network. It comprises multiple convolution operations, each composed of a kernel matrix, usually smaller in spatial dimensionality. These kernels will slide over the entire input image horizontally and vertically. As it moves, the dot product is calculated for each value in that kernel, resulting in a reduced output feature map, as illustrated in Figure 2.3. This way, a pixel in the output feature map is not processed individually and is only connected to a subset of pixels in the input. The size of the step that the kernel slides, known as stride size, is usually 1.

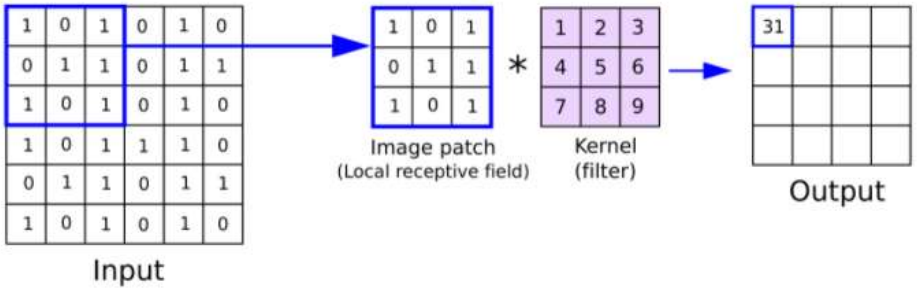


Figure 2.3: Illustration of a convolutional layer

*Left to Right: Input image with the kernel area, image patch, represented as a blue square. Next, the dot product between the image patch and the kernel. Finally, the output feature map with the value resulting from this dot product. The output value is 1*1+1*3+1*5+1*6+1*7+1*9 = 31 (Reynolds, n.d.).*

The resulting output is then passed through an activation function. The most commonly used one is Rectified Linear Unit (ReLU) because it increases non-linearity in CNN.

Since the feature map size is always smaller than the input, zero padding is used to maintain the dimensions of the output matrix. In addition, a layer of zero-value pixels is added around the border of the input, preserving the spatial size and improving performance.

2.4.1.3. Pooling Layer

The pooling layer aims to reduce the dimensionality and, thus, reduce the number of parameters and computational complexity. Furthermore, it is inserted after every convolutional layer with the purpose of making the CNN more invariant to small changes in the input, which helps to extract dominant features (Goodfellow et al., 2016).

The most common type of pooling is max pooling, which uses the maximum value of each portion of the image covered by the feature map. Another type of pooling is average pooling, where instead of using the maximum value, it calculates the mean of the values in the kernel. A pooling layer can be defined through two parameters: the stride and the spatial size. Usually, it is applied kernels of 2*2 with a stride of 2 along the input's dimensions

2.4.1.4. Fully Connected Layer

The fully connected layer will perform the same as a traditional multilayer perceptron neural network by connecting every neuron in the previous layer to every neuron in the next layer. Moreover, it is placed in the end to be used for classification.

It involves flattening that is transforming the entire output from the previous layer into a column vector. In the end, an activation function, usually SoftMax or sigmoid, is used to classify the output.

2.4.2. Transfer Learning

Transfer learning is a technique that is nowadays popular in the field of deep learning. The main idea behind transfer learning is to re-use an already trained model, developed for a specific task, to address a different but related problem. Since CNNs usually require a large amount of training data, transfer learning is usually applied when the new dataset is smaller than the one used to train the pre-trained model (Hussain et al., 2018). Typically, the datasets are similar. So, transfer learning can be used as a feature extractor or fine-tuning.

Feature extraction happens when only the last layer of CNN is changed concerning the new classification. For instance, the basic spatial features learned from the pre-trained data used to classify one dataset are used to classify similar information. For example, a model that learned to identify dogs may have spatial features that can be useful to identify cats.

Fine-tuning would occur when the whole model is re-trained on new data with a low learning rate. This procedure can improve the results by adapting the pre-trained features to the new data.

Overall, transfer learning brings a significant benefit because it reduces the time necessary to train a model.

2.5. R-CNN

Region-based Convolutional Neural Networks (R-CNN) was proposed in 2014 to improve the precision in object detection (Girshick et al., 2014). It aims to detect multiple objects in one input

image using features created from CNNs while generating a high-quality segmentation mask for each instance.

R-CNNs algorithm is shown in Figure 2.4. It begins by using a selective search method to extract region proposals, which groups pixels by color, intensity, or texture for each region, generating around 2000 regions of interest (RoI). Each RoI is then fed into a CNN to generate a dimensional feature vector as output. Afterward, for each output, Support Vector Machine (SVM) classifiers are used to classify the object within that candidate region proposal. Furthermore, the algorithm also predicts offset values to increase the precision of the bounding box by feeding the vector into a bounding box regressor.

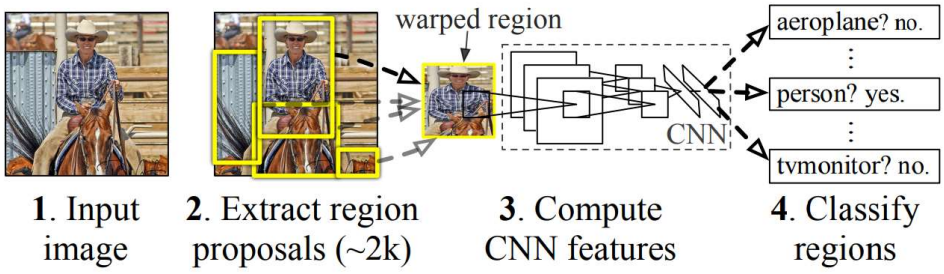


Figure 2.4: Architecture of R-CNN

1. Input image that is given to the algorithm; 2. Extraction of the 2000 region proposals; 3. CNN layer to compute the features for each region; 4. Classification of each region using SVMs (Girshick et al., 2014).

2.6. FAST R-CNN

Fast R-CNN was proposed as an improvement of R-CNN by using RoI pooling since classifying around 2000 region proposals per image would still take a long time to train the network (Girshick, 2015). Instead of feeding every region proposal to CNN, the input image is fed to the CNN only once, providing a convolution feature map. The proposal regions are identified with selective search and reshaped using a RoI pooling layer. Then, these regions are fed to a sequence of fully connected layers that separate into two final layers: a softmax layer to classify the proposal region and a regressor layer to predict the offset values for the bounding box. Figure 2.5 shows the architecture of the algorithm.

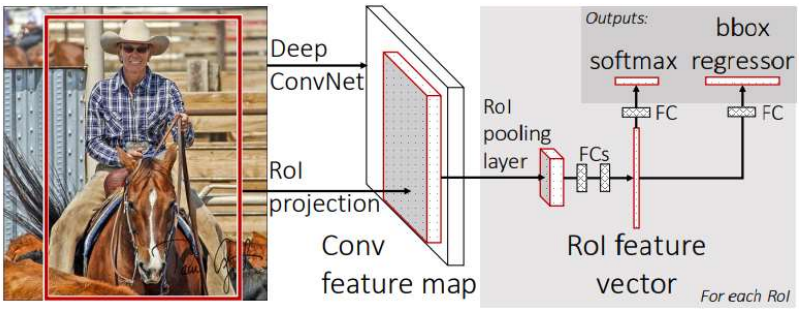


Figure 2.5: Architecture of Fast R-CNN

The input image is fed to a CNN to generate the convolutional feature map. Next, the RoIs generated by the selective search method are projected to the Conv feature maps, which are then fed to a RoI pooling layer to reshape them to the same size. Then, each RoI is mapped to a feature vector by fully connected layers. Finally, the output layers classify each RoI with a classification layer and adjust the bounding boxes with a regression layer (Girshick, 2015).

2.7. FASTER R-CNN

Fast R-CNN also uses selective search to generate the region proposals, which is a time-consuming process. This way, Faster R-CNN was created to eliminate the selective search algorithm using a Region Proposal Network (RPN) (Ren et al., 2016). The algorithm can be visualized in Figure 2.6.

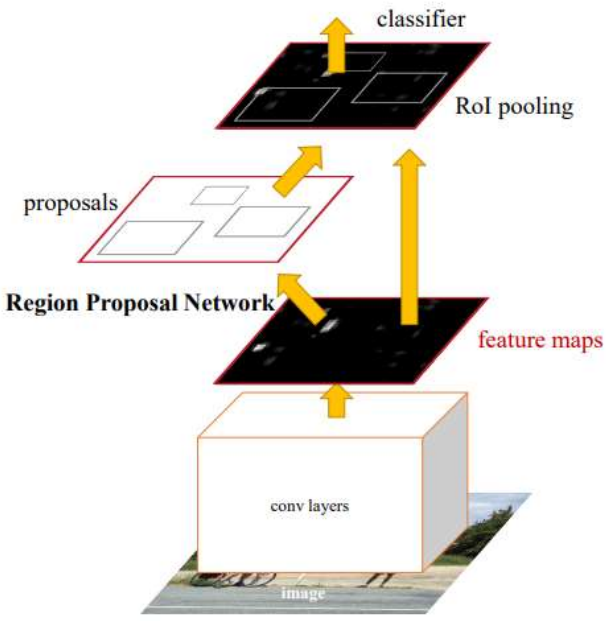


Figure 2.6: Architecture of Faster R-CNN

The input image is fed into a CNN to generate a feature map, which is followed by an RPN to create the anchor boxes. Next, non-maximum suppression is used to reduce the number of RoIs. Afterward, a RoI pooling layer scales each RoI to the same size. Finally, a classification layer classifies each RoI, and a regression layer adjusts the bounding boxes (Ren et al., 2016).

Like Fast R-CNN, the input image is first fed to a CNN to generate a convolutional feature map. However, instead of the selective search, the RPN identifies the region proposals, visualized in Figure 2.7. It works by moving a sliding window through the feature map, creating a set of anchor boxes with three different aspect ratios and three different scales, creating a total of 9 anchors. Then, the network simultaneously uses a classification layer and a regression layer to learn which anchor boxes have an object in them and learn the anchor boxes' offsets in order to adjust for fitting the objects, respectively. The anchor boxes use the Intersection over Union (IoU) score to understand whether it is foreground or background (Ren et al., 2016).

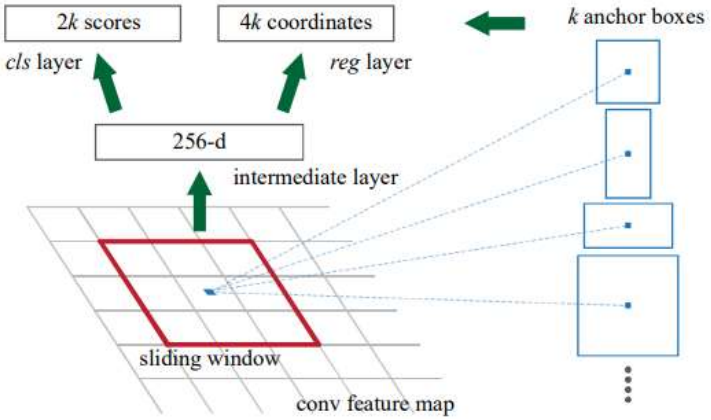


Figure 2.7: Region Proposal Network process (RPN)

For each anchor point of the respective sliding window in the feature map, k anchor boxes are generated (Ren et al., 2016).

2.7.1. Intersection over Union

Intersection over Union (IoU) is calculated by dividing the area of overlap between the predicted bounding box and the ground-truth bounding box by the area of the union of both boxes (Ren et al., 2016). Figure 2.8 shows a representation of this calculation.

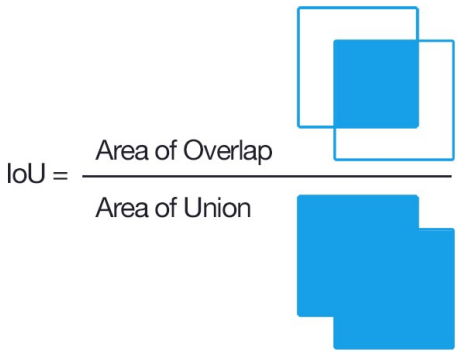


Figure 2.8: Intersection over Union (IoU)

IoU is equal to the area of overlap divided by the area of union (Rosebrock, 2016).

When training the RPN, for each anchor box, it is assigned a positive or negative score based on the IoU. The score can range from 0.0 to 1.0, 0.0 meaning no intersection, and 1.0 when the boxes are equivalent. The anchor receives a positive label when the score is higher than 0.7 with any ground-truth box, but if no anchor has a score higher than 0.7, then the one with the highest value is assigned the positive label. The negative label is when the value is lower than 0.3 for all ground truth boxes. If an anchor box is neither positive nor negative, it does not contribute to the training. Thus, an anchor with a positive label is considered foreground, whereas a negative label is background.

2.7.2. Non-Maximum Suppression

Non-maximum Suppression (NMS) is applied to reduce the number of bounding boxes proposals. It does this by removing the boxes taking into account a specific threshold value (Ren et al., 2016). NMS starts by sorting from high to low all the regions based on their scores. Then, it selects the box with the highest score and computes the IoU between this and all the remaining boxes. The ones that have an IoU greater than the threshold are removed. These steps are repeated until there are no more proposals. Lastly, NMS removes unnecessary bounding boxes, keeping only the best ones, that is, avoiding detecting the same object multiple times (Schmidt et al., 2018). Figure 2.9 shows an example of the before and after the application of NMS.

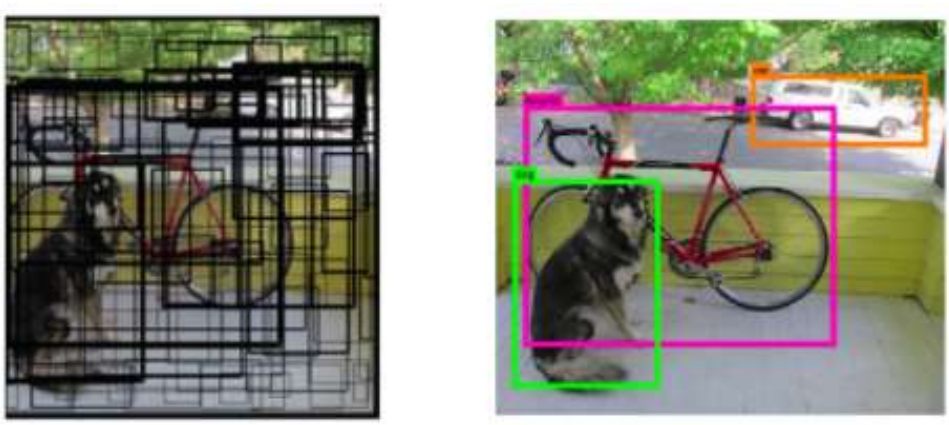


Figure 2.9: Before and after applying Non-Maximum Suppression

A: Before applying NMS, there are multiple bounding boxes; B: Afterward, only the final bounding boxes remain (Singh, 2020).

2.7.3. Balanced Samples

The resulting proposals are sampled to balance positive and negative anchors, particularly having a ratio of up to 1:1. The goal is to avoid bias towards the most dominant label, usually negative (Ren et al., 2016).

2.7.4. RoI Pooling Layer and Final Layer

After NMS, the proposal boxes, the regions of interest (RoI), are fed into a RoI pooling layer to resize the regions, which converts all the regions into the same size. And then, they are fed into a fully connected layer to classify the object and predict the offset values for the bounding box, just like the last step in the Fast R-CNN.

2.8. MASK R-CNN

Mask R-CNN is an extension of Faster R-CNN by adding instance segmentation, a branch to predict segmentation masks on each RoI, in parallel with the classification and bounding box regression branch (He et al., 2018). Additionally, mask R-NN also replaces the RoI pooling layer with a layer called RoIAlign. Figure 2.10 shows a representation of the Mask R-CNN architecture.

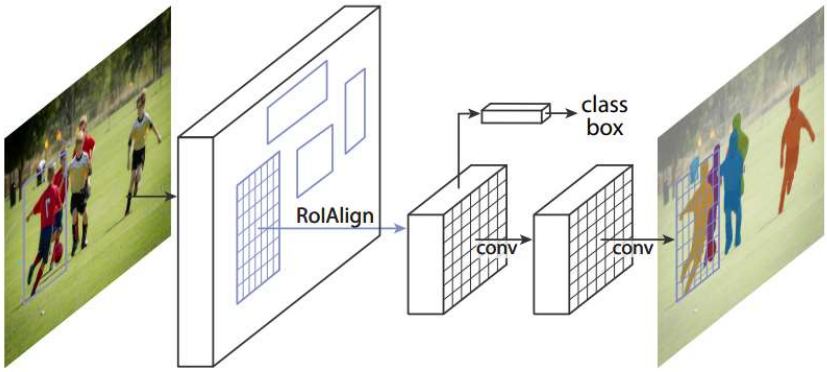


Figure 2.10: Mask R-CNN architecture

It shows the Mask R-CNN framework for instance segmentation. The input image is fed to a CNN to generate a feature map. Next, RPN generates the RoIs, and a RoIAlign layer scales these regions. Then, the scaled RoIs go to classification and bounding box regression layers, in parallel with a segmentation layer (He et al., 2018).

Similar to Fast R-CNN, the input image is fed to a CNN to generate the feature map. Subsequently, an RPN generates multiple RoI using a binary classifier, and an NMS is applied. Then, a RoI align layer is used to scale the RoIs. These are then fed into classification and bounding box regression in parallel with the segmentation layer. That is, mask R-CNN predicts the class and box offset and outputs a binary mask for each RoI, a matrix, where 1s represent pixels that belong to an object and 0s elsewhere (He et al., 2018).

2.8.1. Instance Segmentation

Instance segmentation requires object detection and semantic segmentation. That way, it can correctly detect and segment all objects in an image by assigning a label to each pixel of that image (Li et al., 2017).

2.8.2. Mask Representation

A mask encodes the spatial layout of an object. Thus, it is different from the label and box offsets encodings, which are short vectors. An $m \times m$ mask is predicted from each RoI to allow a consistent object spatial layout. The masks' spatial structure is extracted using the pixel-to-pixel correspondence provided by convolutions, which requires the RoI features to maintain the direct per-pixel spatial correspondence (He et al., 2018).

2.8.3. RoI Align

Image segmentation depends on pixel-level specificity. So, using RoI pooling layers leads to slightly misaligned masks caused by harsh quantization. Therefore, a RoIAlign layer was developed to align the extracted features with the input correctly. RoIAllign uses bilinear interpolation to compute the exact floating values of the features at four different locations in each RoI bin and aggregate the result (He et al., 2018). Figure 2.11 shows the bilinear interpolation for an anchor that is misaligned to the feature map.

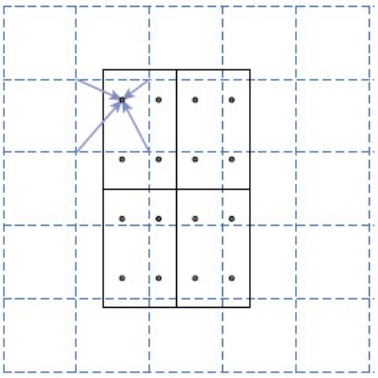


Figure 2.11: RoIAlign

The dashed grid represents a feature map, the solid lines a RoI (2×2 bins), and the dots the four sampling points in each bin (He et al., 2018).

2.8.4. Loss Function

During training, a multi-task loss is defined on each sampled RoI as equation (2.2).

$$L = L_{cls} + L_{box} + L_{mask} \tag{2.2}$$

The classification loss L_{cls} and bounding-box loss L_{box} are the same as in Fast R-CNN. Also, these losses are independent of each other since they are trained and predicted in parallel (He et al., 2018).

In Fast R-CNN, each RoI is labeled with a ground-truth class u and a ground truth bounding box regression target v. A multi-task loss L, equation (2.3), is used on each RoI to train in parallel the classification and bounding-box regression.

$$L(p, u, t^u, v) = L_{cls}(p, u) + \lambda[\mathrm{u} \geq 1] L_{loc}(t^u, v) \tag{2.3}$$

in which

$$L_{cls}(p, u) = -log p_u \tag{2.4}$$

L_{cls} is log loss of the true class u, p is the discrete probability distribution of the categories. The L_{loc} is defined over a tuple of true bounding-box regression targets for class $u, v = (v_x, v_y, v_w, v_h)$ and a predicted tuple $t^u = (t_x^u, t_y^u, t_w^u, t_h^u)$ for the bounding-box regression of the class u. The Iverson bracket indicator function [u ≥ 1] evaluates to 1 when u ≥ 1; 0 otherwise, where '1' is labeled foreground and '0' foreground. The λ controls the balance between the two classes' losses (Girshick, 2015).

The bounding box regression computes four coordinates, defined in equation (2.5), where x and y represent the center coordinates, and w and h the width and height of the box. If one of these letters is followed by a small 'a' it represents the anchor box, and if it is followed by '*' it represents a value for the ground truth box. Thus, this can be described as bounding-box regression from an anchor box to a nearby ground-truth box (Ren et al., 2016).

$$\begin{gathered} t_x = \frac{(x - x_a)}{w_a}, \; t_y = \frac{(y - y_a)}{h_a}, \\ t_w = \log\left(\frac{w}{w_a}\right), t_h = \log\left(\frac{h}{h_a}\right), \\ t_x^* = \frac{(x^* - x_a)}{w_a}, t_y^* = \frac{(y^* - y_a)}{h_a}, \\ t_x^* = \log\left(\frac{w^*}{w_a}\right), \; t_y^* = \log\left(\frac{h^*}{h_a}\right); \end{gathered} \tag{2.5}$$

For the bounding-box regression loss, L_{box}, the equation is defined as equation (2.6).

$$L_{loc}(t^u, v) = \Sigma_{i \in \{x,y,w,h\}} smooth_{L1}(t_i^u - v_i) \tag{2.6}$$

$$smooth_{L1}(x) = \begin{cases} 0.5x^2 & if\ |x| < 1 \\ |x| - 0.5 & otherwise \end{cases}$$

The mask branch generates a mask with m × m dimensions for each RoI, thus having an output with size K.m^2, where K is the number of classes. Since the model is trying to learn a mask for each class, there is no competition between classes. The L_{mask} is defined as the average binary cross-entropy loss, and if the region is associated with the ground truth class k, then L_{mask} only includes the k-th mask (He et al., 2018).

2.9. U-NET

U-Net is a convolutional neural network developed for biomedical image segmentation. It was created in 2015, the same year as Fast R-CNN because the pre-existing CNN focused only on image classification, where the output is only one label. However, in other tasks, like processing biomedical images, it is also required to know the mask's localization (Ronneberger et al., 2015). Therefore, the U-Net is based on a fully convolutional network. It aims to generate a segmentation binary mask, where 1 indicates the foreground, meaning it belongs to a cell, and 0 indicates the background. This mask can be visualized in Figure 2.12.

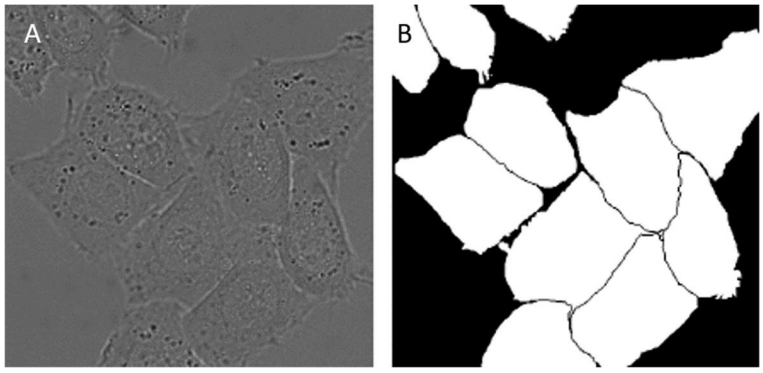


Figure 2.12: U-Net segmentation mask

A: Raw image; B: Segmentation binary mask. Image adapted from (Ronneberger et al., 2015).

The architecture of U-Net is different from all the other R-CNN algorithms, having a u-shape, as can be seen in Figure 2.13, where the left side is the contracting path, and the right side is the expansive path.

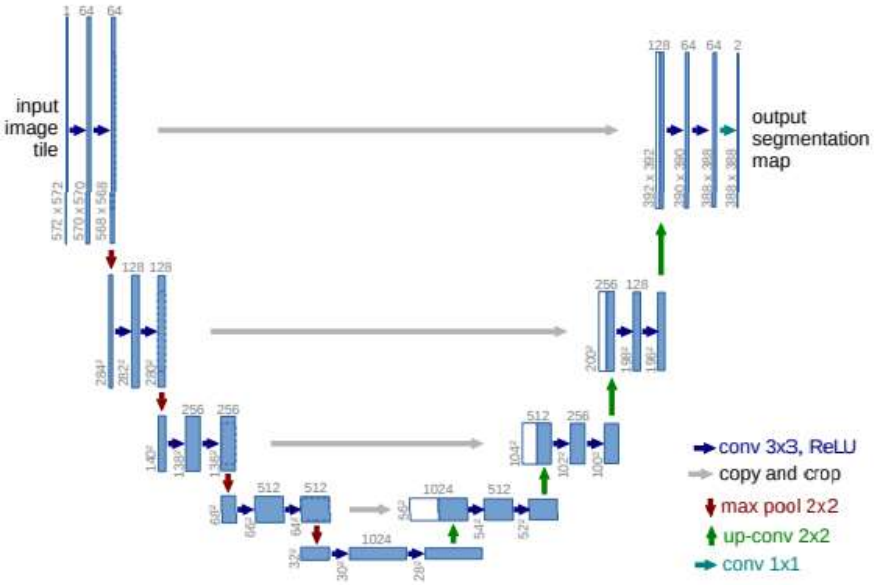


Figure 2.13: U-Net Architecture

The input image is passed fed to convolutional layers in the contractive path to extract the features. Then, the expansive path extracts the exact localization (Ronneberger et al., 2015).

The contractive path, also known as encoder, is similar to the architecture of a traditional CNN. It consists of the repeated application of convolutions, each followed by a ReLu and a max-pooling operation for downsampling. During this process, the spatial information is reduced while feature information is increased. Whereas the expansive path, also known as decoder, is used to compute the exact localization. It is almost symmetrical to the contracting part and combines features and spatial information. It consists of a repeated process of up-convolutions and concatenations of high-resolution features maps from the contracting path (Ronneberger et al., 2015).

One challenge in the segmentation of biological images is separating touching objects, so U-Net uses a weighted loss in the loss function, where the separating background between touching objects receives larger weights (Ronneberger et al., 2015).

3. LITERATURE REVIEW

Over the past few years, there has been a growing interest in the application of Deep Learning to bioimage analysis. As shown in Figure 3.1, the scientific literature on deep learning applied to image analysis and related fields has been increasing exponentially in the last decade (until 2019). Consequently, there has been a significant effort from the biological community to identify the best method for each different study.

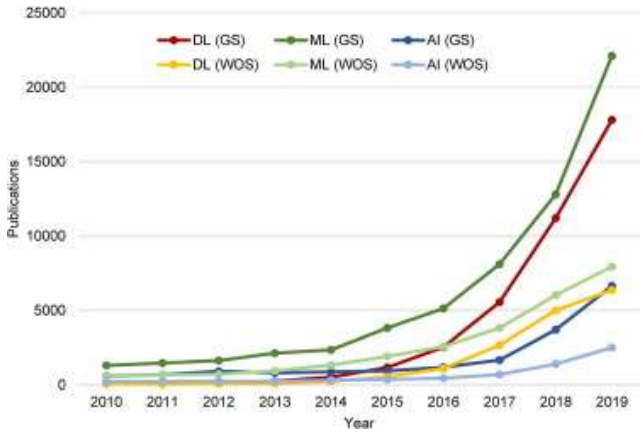


Figure 3.1: The number of scientific publications per year in the last decade (until 2019) on deep learning and related terms

Publications with the terms Deep learning (DL), machine learning (ML), or artificial intelligence (AI) in the title, according to Google Scholar (GS) and Web of Science (WOS). Adapted from (Meijering, 2020).

The interest in applying deep learning to extract valuable information from biological images is growing so much that many implementations have been developed, each one tailored to different aspects of bioimage analysis. This chapter presents the software chosen to achieve the aim of the project and offers an overview of other existing DL and AI-based software developed for bioimage analysis.

3.1. STARDIST

Automatic detection and segmentation of cells is a fundamental procedure in analyzing biological images, and in recent years, many deep learning-based algorithms have been proposed. However, in images with crowded cells, they become susceptible to some segmentation errors. StarDist was proposed in 2018 (Schmidt et al., 2018) to address this limitation. It is a deep learning network able to predict a flexible shape representation using star-convex polygons, which can output a roundish shape, similar to cell nuclei. Moreover, the StarDist algorithm is based on U-Net and is simple to use and train.

StarDist predicts a star-convex polygon for every pixel, meaning that for every pixel with index i, j, it regresses the distances $\{r_{i,j}^k\}_{k=1}^n$ to the boundary of the object with equidistant angles, as can be seen in Figure 3.2.B. In addition, it predicts if a pixel is part of an object, considering only those with a high probability $d_{i,j}$ for polygon proposals. NMS is applied after computing the polygon candidates to retain only the polygons with the highest object probability in a particular region (Schmidt et al., 2018).

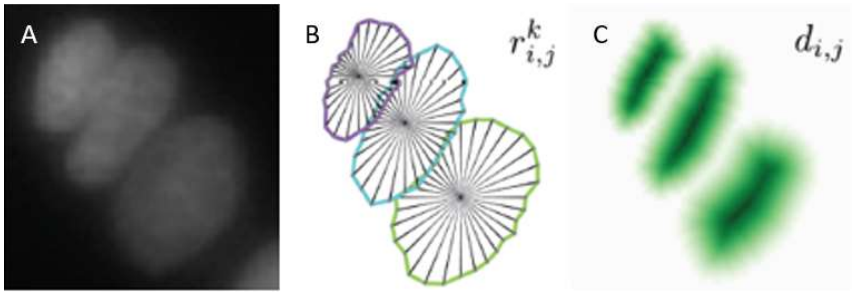


Figure 3.2: StarDist

A: Raw image; B: Star-polygons parameterized by the radial distance $r_{i,j}^k$; C: Predicted objects probabilities $d_{i,j}$; Image Adapted from (Schmidt et al., 2018).

To classify if a pixel is part of an object, StarDist uses an object probability $d_{i,j}$ as the Euclidean distance to the nearest background pixel. The Euclidean distances $r_{i,j}^k$ to the object boundary can be calculated by following each radial direction k until a pixel with a different object identity is encountered. This way, instead of doing binary classification, the NMS favors polygons associated with pixels closer to the center of the cell.

StarDist has as a base the U-Net architecture. However, after the final U-Net feature layer, an additional convolutional layer is added to prevent the two subsequent output layers from fighting over features. This final layer outputs the object probability.

In the paper (Schmidt et al., 2018), StarDist was compared with Mask R-CNN and traditional U-Net. It demonstrated that the first methods based on axis-aligned bounding boxes, mask R-CNN, cannot segment some shapes. Additionally, it was demonstrated that StarDist has better accuracy than the traditional U-Net. Thus, concluding that StarDist outperforms all the other methods.

The StarDist python package already has some pre-trained models incorporated, one of which was trained for nuclei segmentation in fluorescence microscopy 2D single-channel images.

3.2. TRACKMATE

TrackMate is a popular software for automated single-particle tracking in 2D. It starts by detecting the objects in the image and then linking these objects over time. More specifically, first, the automatic detection phase uses a Laplacian of Gaussian filter applied to the image, with a sigma customized to the blob estimated size. Then, the tracker used is based on the Linear Assignment Problem mathematical framework. The spots are first linked frame to frame to build track segments, which afterward can be tuned in several ways, like gap closing and splitting and merging events. Furthermore, it has visualization and analysis tools able to show and evaluate these results (Tinevez et al., 2017).

This software is an interactive Fiji plugin (Fiji is a pre-packaged distribution for performing biological-image analysis). In other words, there are multiple consecutive stages, each corresponding to a step in the tracking process, where the users can choose the optimal algorithm and the best parameters for their type of biological images. In the end, it is able to perform many measurements to analyze the results, including spatial and temporal coordinates of the spots in each track, that is, the spot's position in x and y, and the frame number (Tinevez et al., 2017).

Moreover, it is a highly cited paper, with applications ranging from tracking the fluorescence paint spots on the non-human primate's hand to studying the relationship between mid-lateral cerebellar complex spikes and decision making (Sendhilnathan et al., 2021). It has also been used in developmental and cell biology to track the division cycle of the blastoderm cells and to study microtubule dynamics in the zebrafish embryo (Bernardello et al., 2021).

3.3. STARDIST AND TRACKMATE

In 2020, a pipeline was proposed to automate cell segmentation and tracking in fluorescence and brightfield images by combining StarDist and TrackMate (Jacquemet et al., 2020). The idea is to use StarDist to segment the objects in each frame of a 3D image stack and generate tracking files based on the coordinates of the pixels associated with each final polygon. Then, these tracking files can be used as inputs for TrackMate to track the identified objects over time. Before applying StarDist, the authors trained a StarDist model for their data type, which is necessary if the pre-trained models available were not trained in similar data.

3.4. OTHER APPLICATIONS

Other software and platforms, both commercial and open-source, could be used for segmentation and tracking, such as Imaris, Arivis, EmbryoMiner, DeepCell, 3DeeCellTracker, Ilastik, and CellProfiler Analyst 3.0.

Imaris is a commercial software used for live-cell segmentation, automated tracking, and visualization of multi-channel microscopy datasets of 2D images and 3D time series independently of their size (*Imaris for Cell Biologists - Imaris - Oxford Instruments*, n.d.).

Arivis is another commercially available platform with various tools and modules for processing biology images. Arivis Vision4D is a modular software that deals with 2D, 3D, and 4D images from massive datasets. Moreover, it can perform semi-automated/manual segmentation and cell tracking, among other functionalities (*Image Visualization and Analysis*, n.d.).

EmbryoMiner is an interactive framework for large-scale cell tracking of developing embryos. It was implemented on the open-source data mining toolbox SciXMiner for MATLAB. EmbryoMiner aims to visualize, annotate and analyze vast amounts of cell tracking data (Schott et al., 2018).

DeepCell has a deep learning model for cell tracking (Moen et al., 2019). In addition, DeepCell is a platform containing data and many prediction models (*DeepCell*, n.d.). It has an available python package for single-cell analysis of biological images, enabling the user to apply the pre-existing models.

3DeeCellTracker is a deep learning-based pipeline for segmentation and tracking cells in 3D time-lapse images deforming/moving organs (Wen et al., 2021).

Ilastik is an open-source software for image classification and segmentation based on interactive learning. It allows users to train their random forest classifier by providing the labels (Sommer et al., 2011).

CellProfiler Analyst is an open-source software package used to explore and analyze data interactively. In addition, it includes a feature to train machine learning classifiers (Jones et al., 2008).

However, the multiple software mentioned in this section did not offer what was essential for this project. Imaris and Arivis are commercial software, which can become expensive, and thus, they have restricted access. Additionally, their code is private, making it very difficult to know how it works. Moreover, Imaris' statement of handling data independently of their size might be slightly unrealistic since it cannot usually handle a few hundred GB of data. EmbryoMiner framework did not focus on the tracking algorithm, so it was not a good option for our project, which requires highly accurate applications. DeepCell did not have pre-trained models available on similar data to ours at the time, and so, it was not ready to be used for segmentation or tracking. Likewise, 3DeeCellTracker, Ilastik, and CellProfiler Analyst did not offer pre-trained models.

In conclusion, StarDist and TrackMate seem to be the best combination of software to handle our data and provide accurate results, given the characteristics mentioned before.

4. DATA

The data analyzed in this project are time-lapse series of *in vivo Danionella translucida* embryos. The movies are CZI files generated by Zeiss Lightsheet Z.1 microscope and were provided by the Champalimaud Foundation, Lisbon, Portugal.

4.1. DANIONELLA TRANSLUCIDA CARE

The *Danionella translucida* fish were raised at the Champalimaud Foundation Fish Facility, according to animal handling and experimental procedures approved by the Champalimaud Foundation Ethics Committee and the Portuguese Direcção Geral Veterinária and were performed according to the European Directive 2010/63/EU.

4.2. DATA ACQUISITION

4.2.1. Sample Preparation

The strain used was the *Danionella translucida* transgenic line Tg(elavl3:H2B-GCaMP6s). This line is an unpublished line from Judkewitz Lab(*Judkewitz Lab — Imaging Microscopy Neuroscience Berlin*, n.d.).

To obtain the required embryos for the experiment, *Danionella translucida* eggs were collected and transferred to Petri dishes containing E3 medium, where they were microinjected with Alpha-bungarotoxin mRNA and H2B-mCherry mRNA at the one-cell stage (Wan et al., 2019). Then, they were placed in an incubator until the desired stage for the experiment was reached.

Since chorions affect imaging, the embryos were manually dechorionated, illustrated in Figure 4.1, and were screened for the mCherry signal. Since the two constructs of H2B-mCherry and Alpha-bungarotoxin mRNA were injected together, if the mCherry signal is present, then in principle, the embryos also incorporated the alpha-bungarotoxin construct.

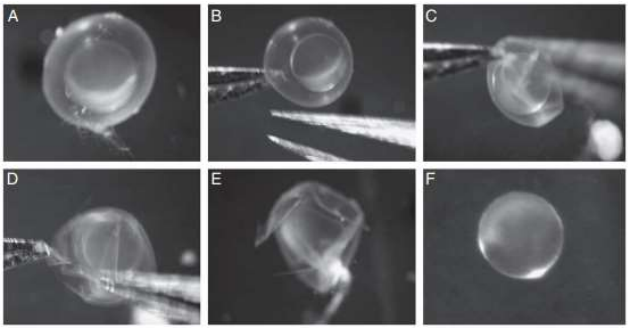


Figure 4.1: Illustrative image of the dechorionation procedure

A: intact embryo; B: egg fixed with one pair of tweezers; C: piercing the chorion with a second pair of tweezers; D: expanding the cleft by pulling apart the chorion using two tweezers; E: chorion with a cleft before lifting; and F: dechorionated embryo. Adapted from (Henn & Braunbeck, 2011).

4.2.2. Light-sheet Imaging

To observe *in vivo* development over time, we used light-sheet fluorescence microscopy, which combines gentle illumination with a high spatial-temporal resolution since only the plane that is being imaged is illuminated. Thus, it enabled us to image Danionella embryos for long periods and with high frequency more efficiently, without inducing significant phototoxicity on the embryos.

Embryos were observed using a Lightsheet Z.1 microscope with one 10x/0.2 illumination objective and one 10x/0.5 detection objective. Solid-state lasers, with an output power of 50 mW, of 488 nm at 25%-35% power and 561nm at 2-5% power, were used to respectively excite GCaMP6f and mCherry fluorophores, with an exposure time of 30ms. A laser blocking filter 405/488/561/640 was used to avoid excitation light detection. A secondary beam splitter LP 560 and emission filters BP 505-545 and BP 575-615 were used to detect the two signals selectively. Detection of emitted fluorescence was realized by two pco.edge 5.5 sCMOS cameras with a pixel size of 6.5 x 6.5 µm, outputting in 16-bit. The acquisition area was set to 800 x 1500 pixels, with a pixel size of 0.455 µm, at an internal zoom factor of 1. The thickness of the light-sheet was 4.12 µm. The temperature throughout the imaging period was maintained at 28.5 °C inside the chamber of the microscope.

For light-sheet imaging preparation, the previously screened *Danionella translucida* embryos were transferred to Eppendorf™ tubes containing 0.8% low melting temperature agarose at around 38 °C. Next, they were drawn into a glass capillary with a 1 mm inner diameter using a Teflon-tipped plunger, inserted into the microscope, which is schematically represented in Figure 4.2.

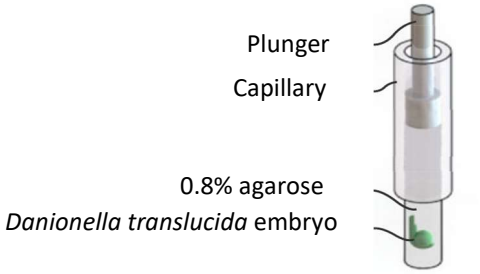


Figure 4.2: Scheme for the preparation for light-sheet imaging

Illustration 0.8% agarose and a Danionella translucida embryo being drawn into a glass capillary (Kaufmann et al., 2012).

The section of the agarose cylinder containing the sample was pulled out and oriented towards the objective lens. At the end of the experiment, the embryo heart rate and overall morphology were visually evaluated, and only the ones with normal heart rates were considered for further analysis.

We acquired volumetric images every 2 seconds for 20 000 cycles, which corresponds to approximately 11.1 hours. A z-stack containing 19 slices with a 5 µm step interval, corresponding to 90 µm of thickness, is acquired in each cycle. The number of slices was chosen as a way to image enough data of the spinal cord but taking into consideration the acquisition frequency.

The time-lapse series were then transformed into maximum intensity projection (MIP), a technique for producing 2D images from 3D data, where it projects the voxels with the highest intensity value on every view throughout the volume onto a plane projection. We used only MIP files because we observed that in this developmental phase, neurons are distributed along the fish spine in a monolayer of cells, as shown in Figure 4.3, which shows the YZ orthogonal projection of an embryo. Therefore, projecting the whole signal in one plane did not reduce the information collected and provided the benefit of reducing the data size by 19 folds. Consequently, the efforts for data handling and computational power were proportionally reduced.

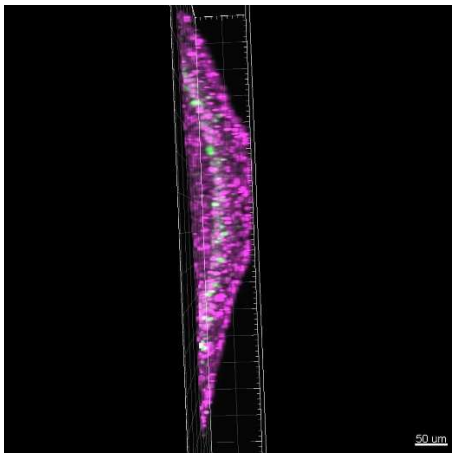


Figure 4.3: YZ orthogonal projection of an embryo

YZ orthogonal projection of a Danionella translucida embryo with the GCaMP6f (green) and the mCherry (magenta) signals.

4.3. DATA DESCRIPTION

The resulting MIP contains 16-bit images of two channels, a green channel corresponding to the GCaMP signal and a magenta to the mCherry signal. Each channel is an image stack composed of 20000 2D images, corresponding to the number of cycles in the experiment. The channels can be visualized in Figure 4.4.

The mCherry fluorescence indicates all the cells in the embryo. Not only is it used to make sure the embryo was injected correctly, as described in section 4.2.1, but it could also be used to identify the orientation of the embryo and to monitor the cells over time. In addition, GCaMP6f fluorescent protein is a calcium indicator for monitoring neuronal activity (Chen et al., 2013).

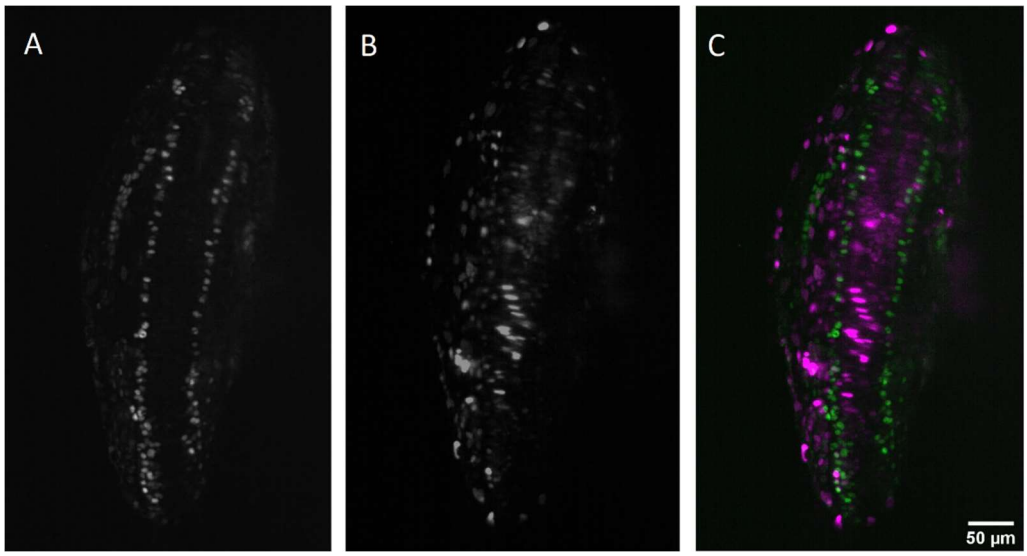


Figure 4.4: Representation of the acquired channels

Channels of one frame of a Danionella translucida embryo. ***A****: GCaMP6f signal;* ***B****: mCherry signal;* ***C:*** *Composite image of A and B with the GCaMP6f (green) and mCherry (magenta) signals overlapped.*

An initial evaluation of the data acquired through this high-frequency observation showed some decrease in the fluorescence intensity. The graph in Figure 4.5 shows the exponential decay of the intensity throughout time. By fitting an exponential regression model in the data, we obtained an R^2 of 0.9816.

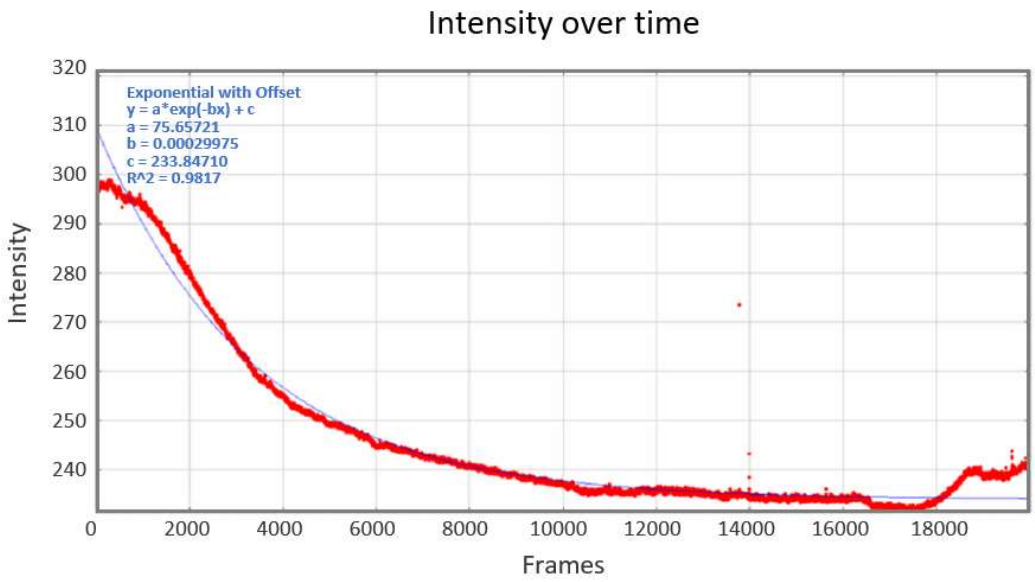


Figure 4.5: Intensity during the time of the experiment

This decay is due to some photodamaging effects caused by prolonged exposure to high-intensity light. Even though the excitation in the light sheet microscope is restricted merely to the volume near the focal plane, there is still some damage to the fluorophores being imaged. Figure 4.6 shows the fade of intensity in the embryo throughout the experiment.

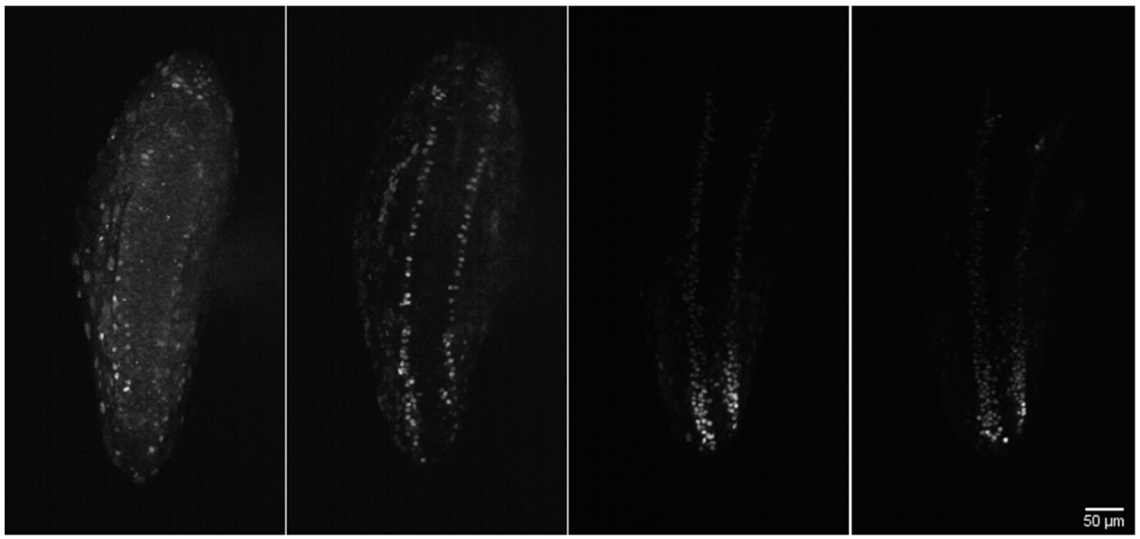


Figure 4.6: Representation of the intensity fading over time in a *Danionella translucida* embryo

Time goes from left to right, where the leftmost image is the first, and the rightmost image is the last.

Since the goal is to understand the neuronal circuit, only the GCaMP6f signal is necessary for the detection, segmentation, tracking, and analysis steps. After further analyzing the images, the active neurons can be observed. They transmit higher intensity light, which can be seen in lighter colors, as

shown in Figure 4.7. On average, the neurons have a 6 µm diameter, and there are around 100-400 neurons detected at each time point.

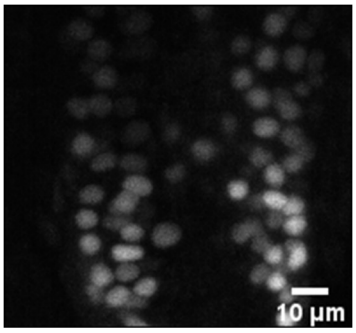


Figure 4.7: Cropped frame from a *Danionella Translucida* embryo

For the next steps of the project, only three movies were considered, all imaged at around the same hours-post-fertilization (hpf) and with similar characteristics. Each movie corresponds to a different embryo.

5. METHODOLOGY

The present chapter describes each step of the analysis process and their respective result to achieve this dissertation's goal. Additionally, the chapter describes the tools used in this project.

5.1. TOOLS

We used Fiji, an open-source platform for biological-image analysis (Schindelin et al., 2012), and the programming language Python 3.7. Fiji was utilized to visualize the data and track the cells with TrackMate (Tinevez et al., 2017). The remaining analysis was performed using Jupyter Notebooks on Anaconda Navigator, a python distribution containing open-source packages for data processing.

5.2. SEGMENTATION USING STARDIST

The first step for the analysis is to segment the neurons. This step was performed using StarDist. As mentioned in section 3.1, StarDist exists as a python package with pre-trained models for 2D, one of which was trained on fluorescence nuclei images, identical to our data. Thus, there is no need to introduce a new model to segment the data since we can predict the segmentation using the model already trained.

The algorithm consists of a loop of repeated steps for each image in the stack. First, a low blurring filter was applied to smooth the image and remove noise. The chosen filter was the median filter with a radius of 2 pixels, given that it efficiently smooths the noise while preserving sharp edges (Justusson, 1981). Next, we used the function predict_instances from the StarDist module to generate a labeled image, the mask, and the star-polygons information, including the outline and center coordinates for each segmented object. The resulting segmentation can be visualized in Figure 5.1. Finally, each labeled image was concatenated into a stack, and it was saved as a single TIFF able to contain several images. Additionally, a RoI folder was exported containing the region coordinates of all segmented objects in each image, which can be directly imported in Fiji. To sum up, StarDist was applied, and its parameters were optimized to get the best possible results.

Different probability and overlap thresholds were tested, and the results were evaluated, by direct observation, to find the optimal one in our data.

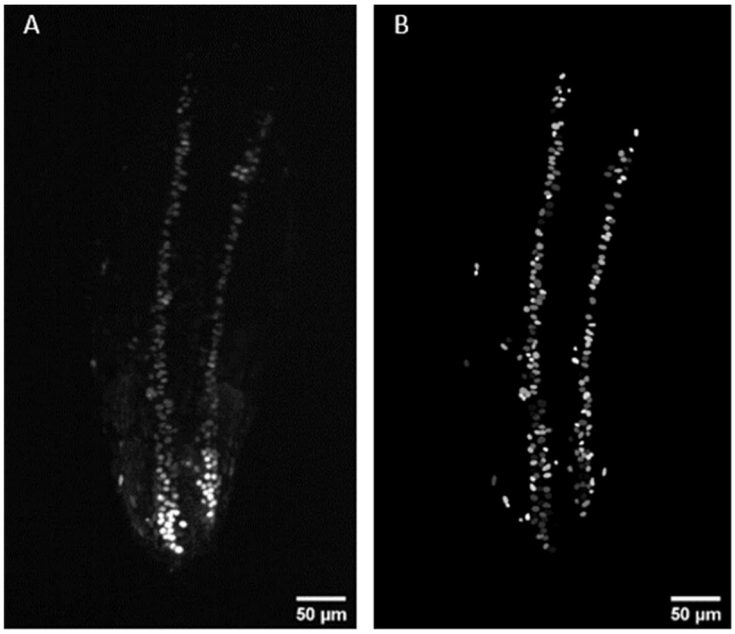


Figure 5.1: StarDist segmentation

A: Original image; B: Labeled image.

5.2.1. Validation

Even though the model used was pre-trained on a similar dataset, it is necessary to guarantee that the segmentation can be considered valid. Hence, to evaluate the segmentation performed by StarDist, two tests were performed. The first one compares StarDist segmentation with human segmentation on a subset of randomly cropped images from different files in our data. However, this test is biased because the absolute number of objects in the image is unknown, and there are discrepancies even between humans in annotating the images.

To overcome this limitation, we generated a set of synthetic data, where the ground truth is known, and compared the performance between the humans and StarDist in segmenting the images. The synthetic images were generated to be similar to the real ones.

In both evaluations, the humans were divided between experts in the biology images (3 people) and non-experts (3 people). Each group was composed of three humans.

5.2.1.1. Comparison between StarDist and human segmentation

For this assessment, thirty images were randomly cropped from our data to contain diverse regions of interest from different frames. The cropped images were small, about 100 × 100 pixels, and each included around twenty to thirty nuclei. Both groups, experts and non-experts, received fifteen

images, and all the humans in that group segmented all fifteen pictures. In addition, the thirty images were also segmented by StarDist.

Before running this test, we compared the performance of StarDist between cropped and entire images to observe if its performance would diminish when cropping an image. A set of randomly selected images was segmented using StarDist, and subsequently, each image was cropped on a specific region of interest. Next, the same set of images were, first, cropped on the same region of interest, and then, StarDist was used again, receiving as input only the cropped area.

Let's consider that group A contains the images, which the segmentation was performed on the entire image first, and group B includes the others. The next step was to compare the number of segmented objects between each duplicate pair. On average, group B had 80% of the total number of items segmented on group A. The fact that group B has a smaller number of segmented objects could be because when cropping the image, some objects on the region's border are also cropped, and thus, StarDist cannot recognize it as an object. However, this can also occur to humans when having access to only part of an image. Therefore, the difference was logical and StarDist performance on cropped images was not considered worse than on the entire frame.

The humans performed the segmentation manually on the Fiji platform. Then, they exported the RoIs of each object so we could compare the coordinates between the segmented objects in an image by StarDist and by a human. Additionally, they segmented the images independently from each other to avoid bias. Figure 5.2 shows the comparison between the segmentation by a human, Figure 5.2.B, and by StarDist, Figure 5.2.C.

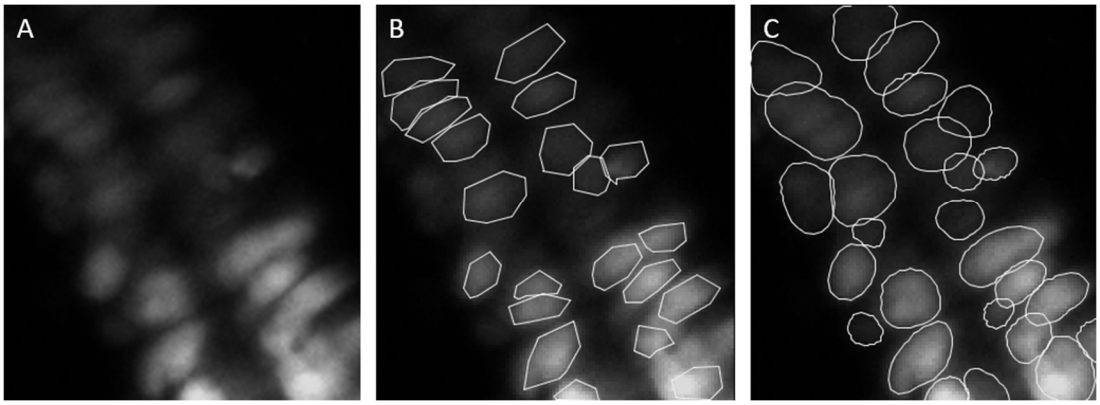


Figure 5.2: Example of the segmentation performed by a human and by StarDist

A: Original image; B: Human segmentation; C: StarDist segmentation.

We started by comparing the number of objects identified. For each human, it was calculated the statistical mean between $\frac{quantity\ of\ RoIs\ from\ StarDist's\ segmentation}{quantity\ of\ RoIs\ from\ a\ human's\ segmentation}$ on each image. On average, StarDist segmented 1.9 times more objects than the humans.

Nonetheless, the quantity of objects doesn't inform if the objects segmented are the same between humans and StarDist. For example, it may happen that StarDist and a human segmented the same number of items in an image, but they are different. For this purpose, it was verified how many segmented neurons were the same by comparing the position of the centroids of each. The centroids were calculated from the RoI coordinates. However, even though all the humans used the Fiji platform for the segmentation, they chose different selection tools, implying that the RoIs between some humans had the information organized differently. In total, there were three types:

- Two lists contained coordinates corresponding to the outline of the RoI, one for the x-axis and another for the y-axis. The centroid could be calculated by the average point, that is $(x_c, y_c) = (avg(\sum x), avg(\sum y))$;
- Values for top, left, width, and height, i.e., top, correspond to the y lowest value since the y axis is from top to bottom on Fiji. The centroid was $(x_c, y_c) = (left + \frac{width}{2}, top + \frac{height}{2})$;
- Coordinates of two points, $(x1, y1)$ and $(x2, y2)$, corresponding to the extremes of a line segment in the RoI. The centroid is the center of that segment, $(x_c, y_c) = \left(\frac{x1+x2}{2}, \frac{y1+y2}{2}\right)$.

Two objects are considered the same if the distance between their centroids is lower or equal to 3 pixels. This criterion was based on the distribution of the minimum distance between each centroid. The majority had a space lower or equal to 3 pixels, as can be visualized in Figure 5.3.

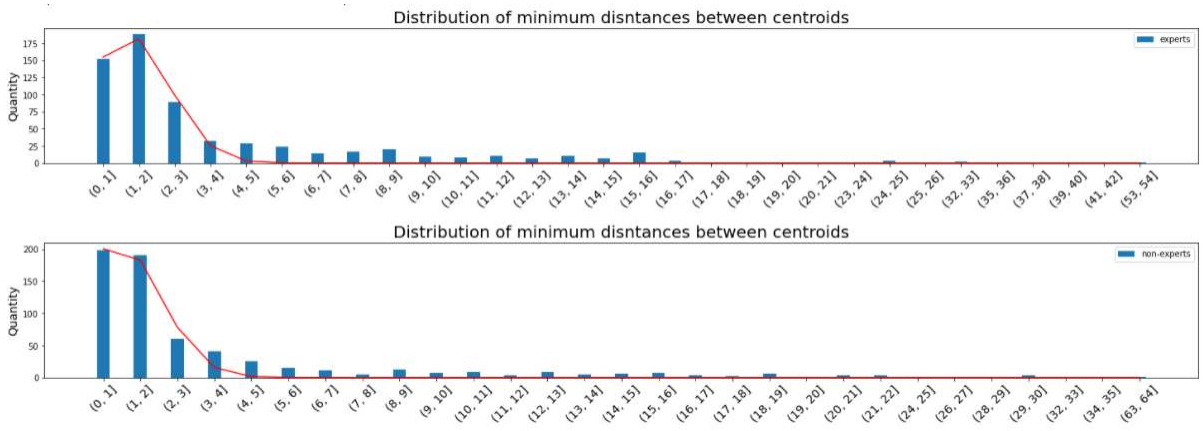


Figure 5.3: Distribution graph for the minimum distance between two centroids

The distances were separated into bins, which are represented in blue color. For experts, 66% of the minimum distances were lower or equal to 3 pixels, and for non-experts was 71%. In addition, the red line is a curve fitting of a gaussian distribution. The R-squared was 0.95 and 0.96 for experts and non-experts, respectively.

The final step is to calculate the percentage of objects identified equally in each image by dividing the number of matching items by the total number from the one that segmented less. On average, 76% of objects were considered the same between humans and StarDist segmentation. In particular, 75% for the experts' segmentation and 77% for the non-experts. To conclude, non-experts' segmentation was more similar to StarDist than the non-experts' segmentation. However, this difference is minimal.

5.2.1.2. Comparison between StarDist and human segmentation performed on synthetic data

Fifteen synthetic images were created on python by creating ellipses of different shapes, sizes, and intensities to be the most possible similar to the actual data. In addition, the ellipses had diameters identical to the neurons, and they were placed randomly in an image with a black background. The number of ellipses created in each image was generated based on the frequency of the number of objects segmented in the thirty previous pictures. In the end, some noise was added to become more realistic. Finally, the synthetic images created were all segmented by StarDist and all the Humans (experts and non-experts). Figure 5.4 shows two of those fifteen images.

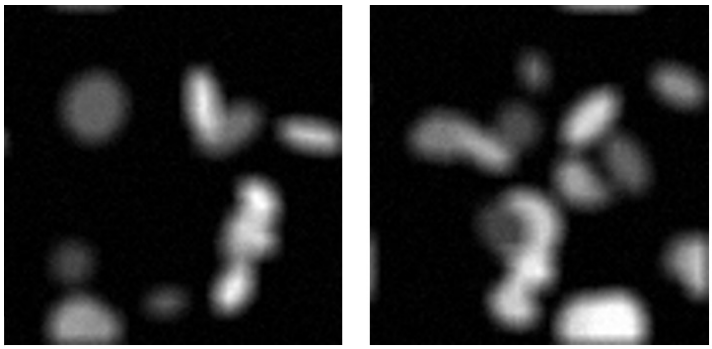


Figure 5.4: Synthetic images

This second evaluation assumed as ground truth the number of objects as returned by the image generating algorithm. Then it was compared with the quantity that StarDist and the humans segmented. Figure 5.5 shows the comparison for each image sorted by the number of objects between the ground truth, StarDist, experts, and non-experts. As the number of items increases, both StarDist and humans segment less, proportionally to the ground truth. This difference is likely due to the presence of overlapping objects in a more populated image. In general, both the humans and StarDist segmented less quantity than the ground truth. On average, StarDist segmented 82% of the ground truth, experts 84%, and non-experts 78%.

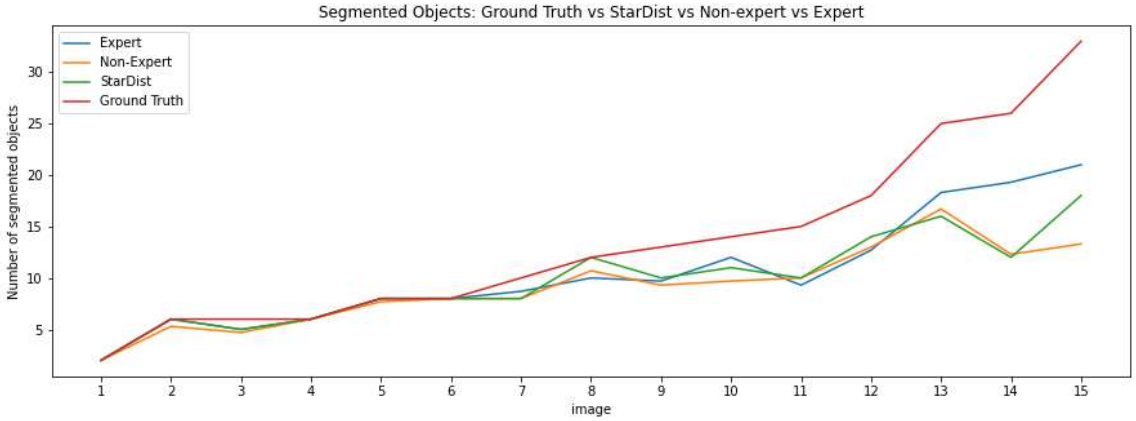


Figure 5.5: Graph comparing the number of segmented objects between StarDist, Experts, and Non-experts on the synthetic images

In Figure 5.6, all the humans are compiled in one line, and it shows that all the humans, on average, perform pretty similar to StarDist. In total, all the humans segmented 81% of the ground truth.

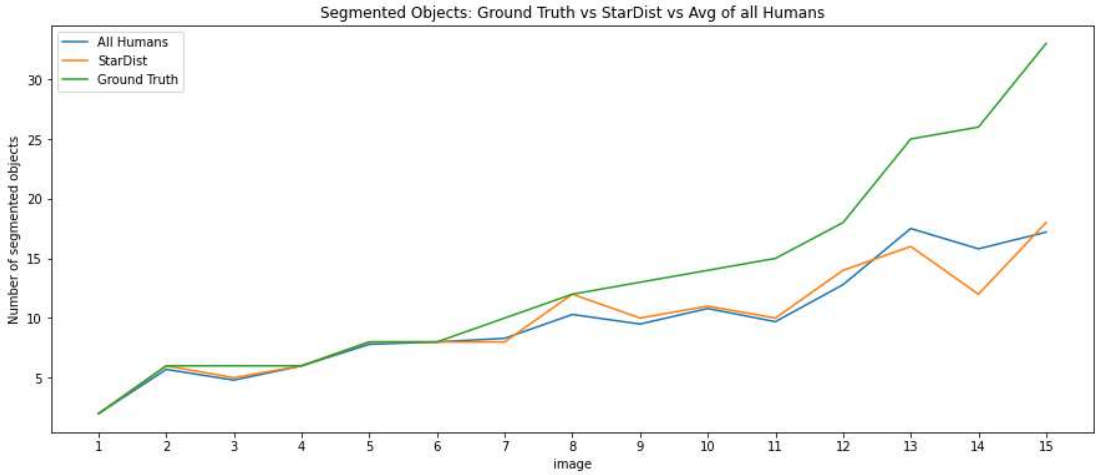


Figure 5.6: Graph comparing the number of segmented objects between StarDist and humans on the synthetic images

5.2.1.3. Conclusion on the validation

To sum up, when comparing the ground truth, humans and StarDist segmentation, it showed that StarDist and humans have almost the same performance, wherein general experts segmented a little more and non-expert a little less than StarDist. Additionally, comparing the StarDist segmentation and the humans segmentation in the actual data showed that StarDist segmented, on average, more objects, and that 76% of the objects segmented between StarDist and the humans were considered the same. Therefore, the segmentation performed by StarDist can be deemed valid for the segmentation step.

5.3. INTENSITIES

To study the neural activity of the *Danionella translucida* embryo, we needed to measure the intensities of the neurons over time. Therefore, after the segmentation performed by StarDist, we calculated for each segmented object in all the images their mean intensities.

The python package scikit-image was developed for image processing (Van Der Walt et al., 2014). It includes a function called 'measure.regionprops_table' able to compute properties for each segmented region. This function receives the labeled image and the original image as input, and it returns a table with the chosen properties, the value of the mean intensity, and the centroid coordinates. Thus, every image generated a table, like Table 5.1, containing the mean intensity, centroid coordinates, and frame number, that is, the position of the image in the stack.

	label	mean_intensity	Y	X	Frame
0	1	375.248408	1263.872611	319.974522	0
1	2	339.176287	765.753510	232.221529	0
2	3	339.477273	915.802273	233.965909	0
3	4	340.740741	691.829630	234.829630	0
4	5	515.142857	227.102041	483.040816	0
...	...	...	...	...	...
238	239	250.098361	599.229508	481.983607	19999
239	240	240.400000	464.523077	369.769231	19999
240	241	314.611111	1012.722222	459.736111	19999
241	242	257.619048	777.857143	461.952381	19999
242	243	243.464286	883.535714	422.107143	19999

4382562 rows × 5 columns

Table 5.1: Intensity table

This table was from the dataset with 20000 frames or cycles. It has five columns, label identity, mean intensity, y coordinate, x coordinate, and frame number.

5.4. TRACKING FILE

In the paper (Jacquemet et al., 2020), the tracking file was generated with the star-polygons center coordinates from StarDist. However, there is a discrepancy between the StarDist's star-polygons centers, and the geometric center retrieved from regionprops, as shown in figure 5.7.

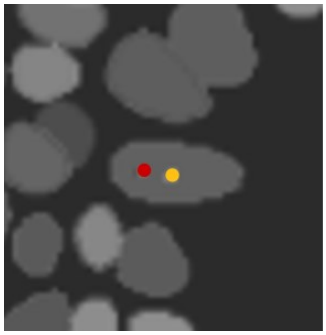


Figure 5.7: Comparison between the centroid from StarDist and the actual geometric center

The red dot is the centroid of the star-polygons resulting from the segmentation by StarDist. The yellow dot is the geometric center, which corresponds to the values on the intensity table.

Therefore, the tracking file was generated using the centroid coordinates from the intensity table. First, all the centroid coordinated were saved organized in lists, each for each image in the stack.

Next, for each position, a point was generated in the corresponding coordinates and frame number. Figure 5.8 shows the code used to create this file. In addition, Figure 5.9 shows an example of the tracking file overlapping the original image.

```
#1. Create a list with lists corresponding the the frames and containing all the point coordinates
points = []
for t in tqdm(range(n_timepoint)):
    df = df_int[df_int.Frame == t]
    l_t = []
    for i, r in df.iterrows():
        l_t.append((round(r.Y),round(r.X)))
    points.append(l_t)

#2. Create the tracking file
Tracking_stack = np.zeros((n_timepoint, shape[2], shape[1]))

for t in tqdm(range(n_timepoints)):
    #list of the points in timepoint t
    points_t = points[t]
    # Create a tracking file for trackmate
    for point in points_t:
          cv2.circle(Tracking_stack[t],point,0,(1), -1)

Tracking_stack_32 = img_as_float32(Tracking_stack, force_copy=False)
Tracking_stack_8 = img_as_ubyte(Tracking_stack_32, force_copy=True)
Tracking_stack_8_rot = np.rot90(Tracking_stack_8, axes=(1,2))
Tracking_stack_8_rot_flip = np.fliplr(Tracking_stack_8_rot)

imsave(Results_folder +'/'+ exp_name +'_tracking_file.tif', Tracking_stack_8_rot_flip, compress=ZIP_DEFLATED)
```

Figure 5.8: Tracking file code

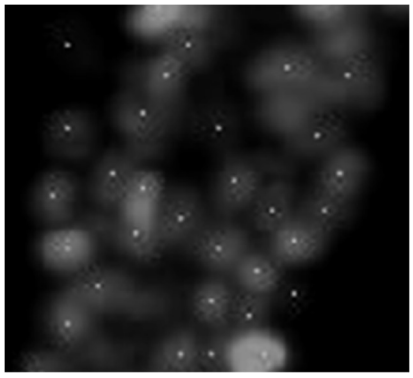


Figure 5.9: Centroids from the tracking file overlapping the original image

This figure is a cropped image from a random frame. The white points are the circles created in the tracking file, and the ellipse shape objects correspond to objects in the original file.

5.5. TRACKING USING TRACKMATE

The tracking file was imported in the Fiji plugin, TrackMate, to connect the dots in time to create the tracks, as can be visualized in Figure 5.10. Trackmate's pipeline main steps begin by choosing a detector. In our case, we used the Downsample LoG detector, with an input size of 1 micron for the

estimated diameter of the objects. The next important step is choosing the tracker algorithm and its parameters. We chose LAP Tracker since it allows us to deal with gap-closing events, that is, having spots linking with a frame gap, because there could be frames where StarDist misses to segment some object. For its parameters, the following values were assigned:

1. The maximum distance between the frame to frame linking was 6 microns, corresponding to the average diameter size of a neuron.
2. The gap closing maximum distance was 10 microns, selected taking into account the maximum distance a neuron can travel.
3. The gap closing maximum frame gap was 400, corresponding to around 13 minutes, which is a small interval in the entire experiment.

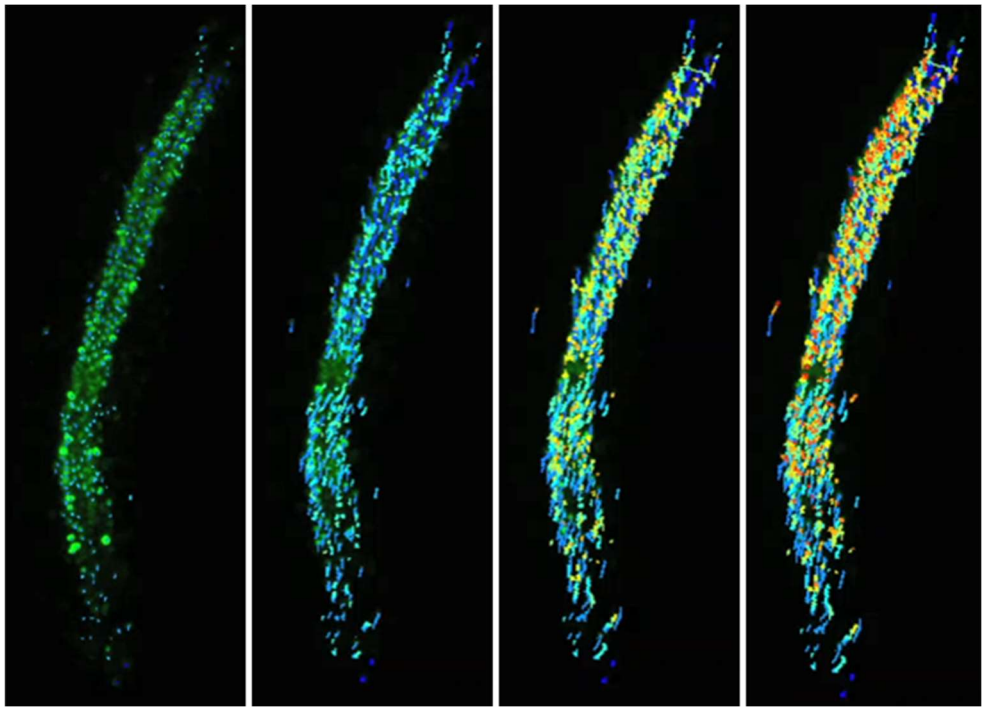


Figure 5.10: Tracks in TrackMate

It is an example of a sequence of images showing the tracks being generated in time.

TrackMate generated tables with information about the tracks. Each spot is assigned to a unique track, identified by a Track ID, and returns its coordinates and frame number. An example is shown in Table 5.2.

	Label	ID	TRACK_ID	QUALITY	POSITION_X	POSITION_Y	FRAME
0	ID1025	1025	0	10389.634	418	641	0
1	ID401	401	0	10389.634	418	640	1
2	ID692	692	0	10389.634	418	640	2
3	ID98	98	0	10389.634	418	640	3
4	ID1293	1293	0	10389.634	418	639	4
...	...	...	...	...	...	...	...
514119	ID466404	466404	1044	10389.635	594	276	1521
514120	ID466694	466694	1044	10389.635	594	276	1522
514121	ID467842	467842	1044	10389.635	594	276	1526
514122	ID468135	468135	1044	10389.634	595	275	1527
514123	ID468418	468418	1044	10389.635	594	276	1528

514124 rows × 7 columns

Table 5.2: TrackMate table – Statistics about spots in tracks

This table has information about the dot coordinates and frame number, the same as the Intensity table. The only difference here is that the coordinates are rounded.

As shown in Table 5.2, for each experiment, more than one thousand tracks were generated.

5.6. TABLE WITH SPOTS INFORMATION AND INTENSITIES

The mean intensities previously calculated (see section 5.3) can be added to the TrackMate Table to the correspondent centroid coordinates and frame number. Each row corresponds to one segmented object. Hence, for each row on the Intensity Table, the coordinates and frame values were first extracted. Then, a row on the TrackMate Table matching those values was searched, and when found, the mean intensity was added in a new column. Accordingly, the final table looks like Table 5.3.

	Label	ID	TRACK_ID	QUALITY	POSITION_X	POSITION_Y	FRAME	mean_intensity
0	ID1025	1025	0	10389.634	418	641	0	7655.644444
1	ID401	401	0	10389.634	418	640	1	7342.164706
2	ID692	692	0	10389.634	418	640	2	5608.969697
3	ID98	98	0	10389.634	418	640	3	4652.071429
4	ID1293	1293	0	10389.634	418	639	4	3968.677966
...	...	...	...	...	...	...	...	...
514119	ID466404	466404	1044	10389.635	594	276	1521	453.781818
514120	ID466694	466694	1044	10389.635	594	276	1522	475.3
514121	ID467842	467842	1044	10389.635	594	276	1526	452.301887
514122	ID468135	468135	1044	10389.634	595	275	1527	463.176471
514123	ID468418	468418	1044	10389.635	594	276	1528	466.297872

514124 rows × 8 columns

Table 5.3: TrackMate table with mean intensity for each object

This table is equal to table 5.2 with an added column for the corresponding mean intensities.

5.7. Distinguish Between Left and Right Side of the Embryo

In order to achieve the goal of describing the neural activity of a Danionella translucida embryo, a piece of vital information is to know if a neuron belongs to the left or right side of the embryo. For this reason, we first needed to have a comparison point, so having the coordinates of the embryo's skeleton allowed us to know if a neuron is on the left or the right. The python package scikit-image (Van Der Walt et al., 2014) contains a function, skeletonize, that computes the skeleton of the object in an image. It works by making successive passes. On each border, pixels are identified and removed if they do not break the connectivity of the corresponding object (*Skeletonize — Skimage v0.19.0.Dev0 Docs*, n.d.). However, the input image has to be binary. Thus, before using the function skeletonize, a blurring filter was applied to smooth the whole embryo, and then a binary mask was created. In Figure 5.11 it can be visualized the steps to produce the skeleton and the result.

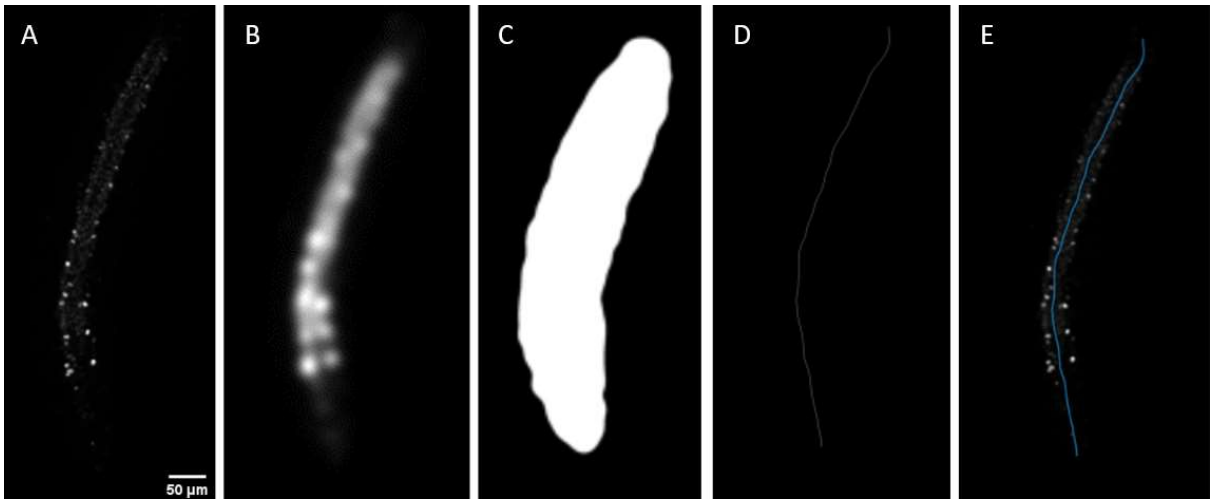


Figure 5.11: Skeleton of an embryo

Example of the process to create the skeleton in an image. A: Original Image; B: Blurred object; C: Binary mask of the blurred object; D: Skeleton generated by the skeletonize function; E: Skeleton, blue line, overlapping the original image.

The pipeline to assign the position of a neuron starts by generating the skeleton for each image, followed by computing the side belonging to each object and adding this information to the TrackMate with Intensities Table, as visualized in Table 5.4. In the cases where the spot belongs to the skeleton coordinates, the side assigned on the table was 'on the line'.

	Label	ID	TRACK_ID	QUALITY	POSITION_X	POSITION_Y	FRAME	mean_intensity	side
0	ID1025	1025	0	10389.634	418	641	0	7655.644444	left
1	ID401	401	0	10389.634	418	640	1	7342.164706	left
2	ID692	692	0	10389.634	418	640	2	5608.969697	left
3	ID98	98	0	10389.634	418	640	3	4652.071429	left
4	ID1293	1293	0	10389.634	418	639	4	3968.677966	left
...	...	...	...	...	...	...	...	...	...
514119	ID466404	466404	1044	10389.635	594	276	1521	453.781818	left
514120	ID466694	466694	1044	10389.635	594	276	1522	475.3	on the line
514121	ID467842	467842	1044	10389.635	594	276	1526	452.301887	left
514122	ID468135	468135	1044	10389.634	595	275	1527	463.176471	on the line
514123	ID468418	468418	1044	10389.635	594	276	1528	466.297872	on the line

514124 rows × 9 columns

Table 5.4: TrackMate table with mean intensities and side information

The side information can have three labels: 'left', 'right', or 'on the line'.

Since the generation of the skeleton is not precise and considering that the neurons move in the embryo over time, spots in the tracks may have different sides assigned over time. To overcome this issue, each track ID, corresponding to a single neuron, was counted which side appeared the most through time, and the majority side was assigned to the track ID, as can be visualized in Table 5.5.

	TRACK_ID	side
0	0.0	left
1	1.0	left
2	2.0	right
3	3.0	left
4	4.0	left
...	...	...
908	1040.0	right
909	1041.0	left
910	1042.0	right
911	1043.0	right
912	1044.0	left

913 rows × 2 columns

Table 5.5: Table with side information for each neuron

5.8. ANALYSIS OF THE TABLES

To understand how the neural activity in each side of the embryo was behaving over time, the TrackMate table with the intensities, Table 5.3, was split into 2, one for each side. Table 5.5.

For each side table, the intensities of all the tracks over time were plotted on a graph. Then, for each track, the intensities over time were standardized using equation 5.1 to rescale the distribution of the values to have a mean equal to 0 and a standard deviation equal to 1.

$$t_{norm} = \frac{(t - mean)}{standard\ deviation} \tag{5.1}$$

Where $mean = \frac{\sum_i^n x}{n}$, and $standard\ deviation = \sqrt{\frac{\sum_i^n (x_i - mean)^2}{n-1}}$, n = number of data points.

Afterward, the tracks were organized according to their y position to have the tracks arranged from tail to head, as shown in Figure 5.12.

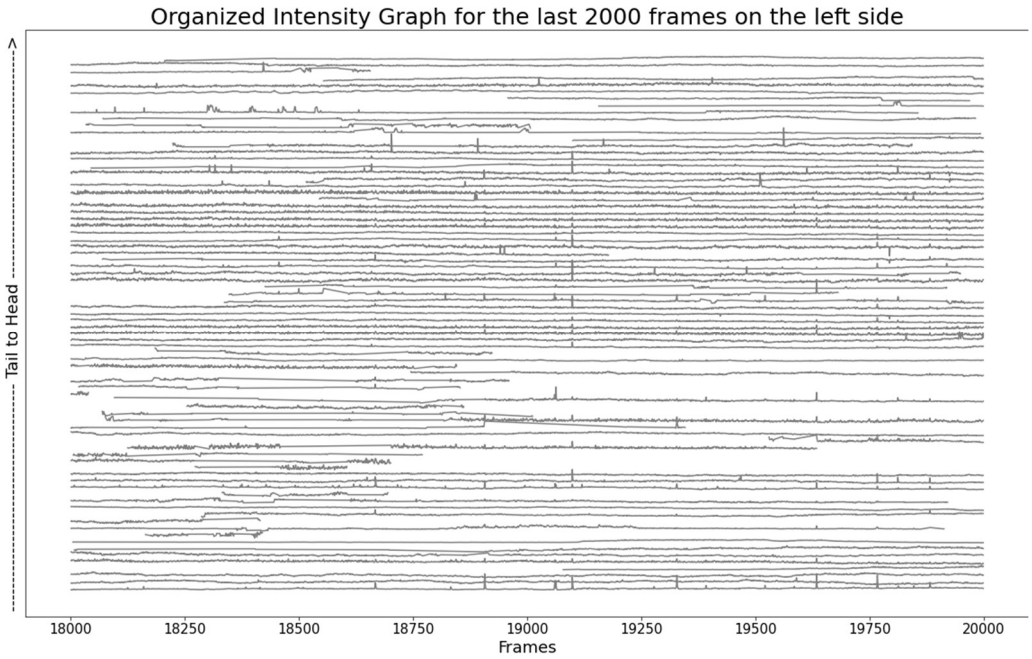


Figure 5.12: Organized graph - Intensity over time

This graph only plotted the last ≈1.1 hours of an experiment and only for the left side of the embryo. Since there are hundreds of tracks, it would not be a clear visualization to plot them all.

For each side, the intensity peaks, which correspond to the firing of a neuron, were studied using the function 'find_peaks' (*Scipy.Signal.Find_peaks — SciPy v1.7.1 Manual*, n.d.) This function belongs to the module Signal of the python library SciPy, developed for scientific computing (Virtanen et al., n.d.). It enabled us to find peaks based on their prominence. The prominence of a peak measures how much the peak stands out by the vertical distance between the highest point and its lowest contour line. This way, we could exclude smaller peaks, which could be related to changes in the embryo's intensity due to the experiment's external factors. After identifying the peaks, their frequency was calculated over time by creating a binary table, where 1 corresponds to the existence of a peak and 0 otherwise (Table 5.6).

TRACK_ID	0	1	2	3	4	5	6	7	8	9	...	19990	19991	19992	19993	19994	19995	19996	19997	19998	19999
4	0	1	0	0	1	0	1	0	0	0	...	0	0	0	0	0	0	0	0	0	0
14	0	1	0	0	0	1	0	1	0	0	...	0	0	0	0	0	0	0	0	0	0
19	0	1	0	0	0	0	0	0	0	1	...	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
24	0	1	0	0	1	0	1	0	1	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
5520	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	1	0	0
5521	0	0	0	0	0	0	0	0	0	0	...	0	1	0	0	0	0	1	0	0	0
5523	0	0	0	0	0	0	0	0	0	0	...	0	1	0	1	0	0	0	1	0	0
5528	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
5529	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

1716 rows × 20000 columns

Table 5.6: Binary table – Existence of peaks

Subsequently, we selected all the peaks from the frames that contained at least two. Then, the y position of those peaks was extracted. Finally, the y position of the peaks was plotted over time, as shown in Figure 5.13.

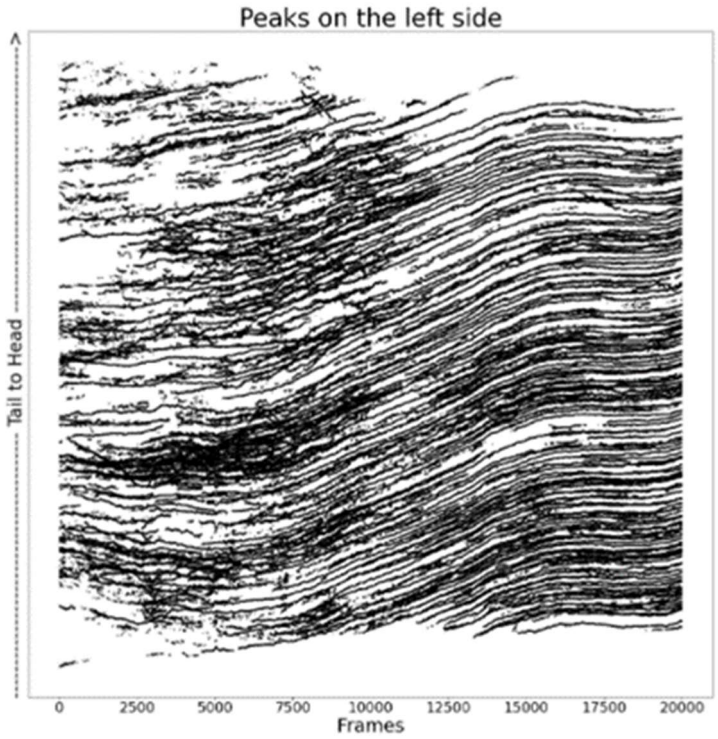


Figure 5.13: Graph with intensity peaks

In this graph, the peaks of all frames were plotted. The x-axis represents the frames, and the y-axis represents the peaks' position in y.

6. RESULTS

The present chapter provides the analysis results on the three movies selected for this study. One of the analysis steps was to separate the embryo into two parts, left and right, which means there are always two final graphs for each movie. The two graphs represent the frequency of the intensity peaks over time and the intensity peaks' position in y over time. They were considered the most important because their visualization gives an overview of the neural activity present in the embryos.

This chapter also shows a graph resulting from a small analysis to an additional subset from a fourth movie. The embryo in this movie was imaged at a later embryonic stage when we can observe the coordinated neural activity. We analyzed a subset of coordinated neurons and plotted their intensities over time.

6.1. NEURAL ACTIVITY OF THE EMBRYOS FROM THE SELECTED MOVIES

6.1.1. Frequency of the Intensity peaks

The frequency of the intensity peaks over time enables us to understand how the quantity evolves over time, whether there was a decrease, increase, or even if they remained stable. Since each movie has 20 000 frames, the peaks were added up in bins of 200 frames to allow a better visualization of the final graphs. Figure 6.1 shows all the frequency graphs for the three movies. Each row corresponds to a different MIP file, and the columns correspond to the side of the embryo in that movie, i.e., the column on the left represents the left side of the embryo, and the right column represents the right side. Each frequency graph has the bins of frames on the x-axis and the quantity on the y-axis.

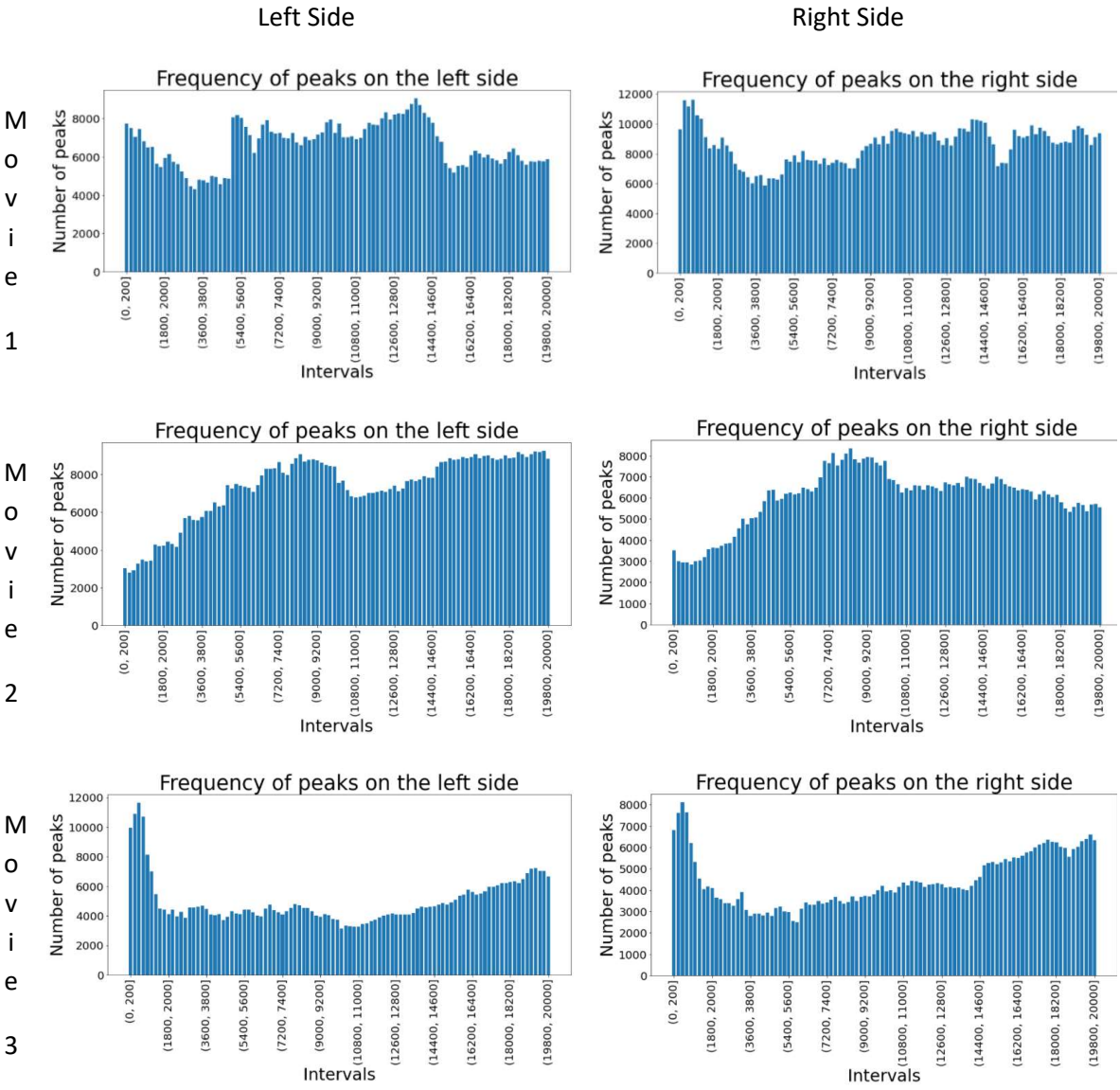


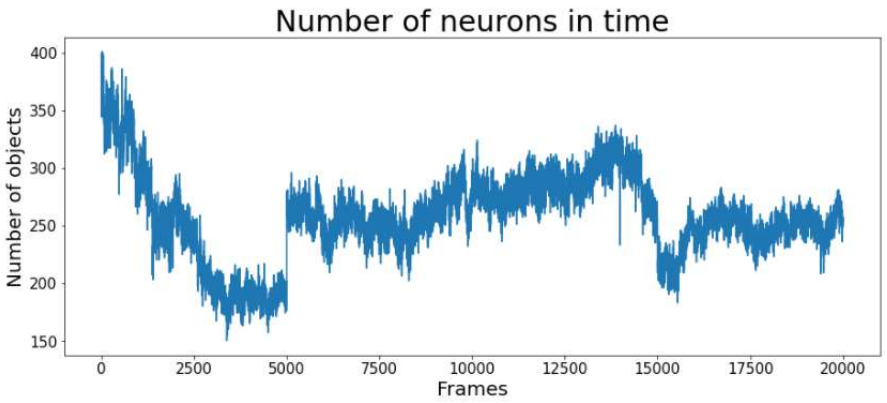
Figure 6.1: Peaks Frequency

The six graphs show the peaks frequency by movie on each side. The rows represent different movies, and the columns correspond to the sides. The column on the left symbolizes the left side and the column on the right, the right side.

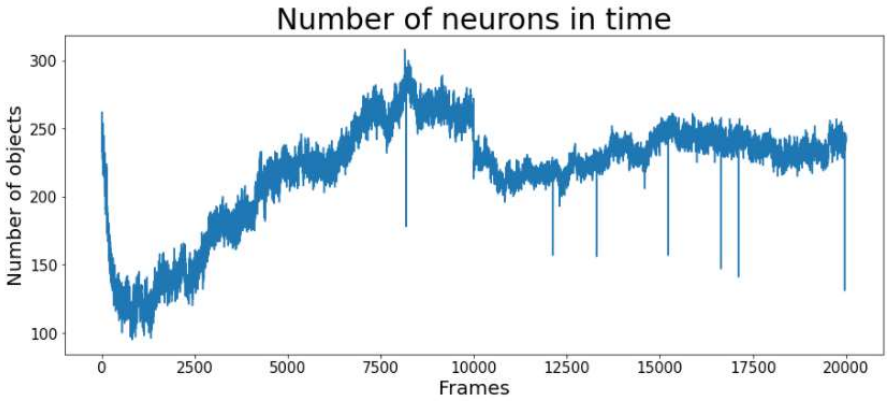
The number of peaks was added up in bins of 200 frames to have a better visualization. On the x-axis, there are the bins, and in the y-axis, the number of peaks.

To understand how the changes in frequency were affected by the total number of neurons in the system, we also plotted the number of all identified objects in each frame. Figure 6.2 shows a graph for each movie with the number of neurons in time.

Movie 1



Movie 2



Movie 3

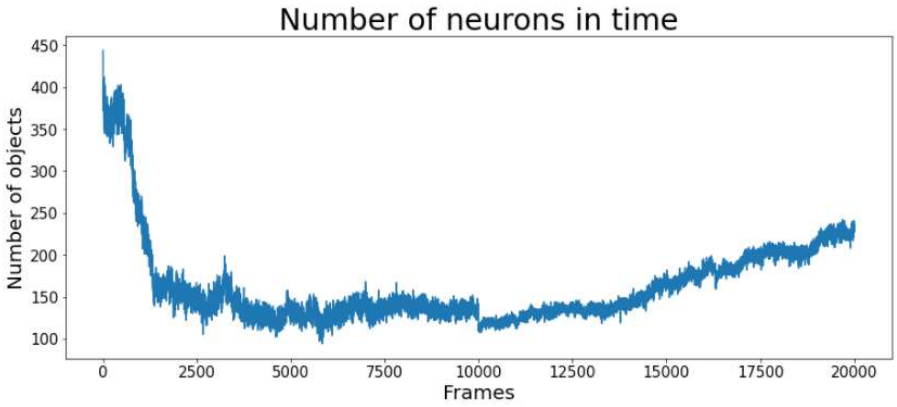


Figure 6.2: Quantity of objects in time

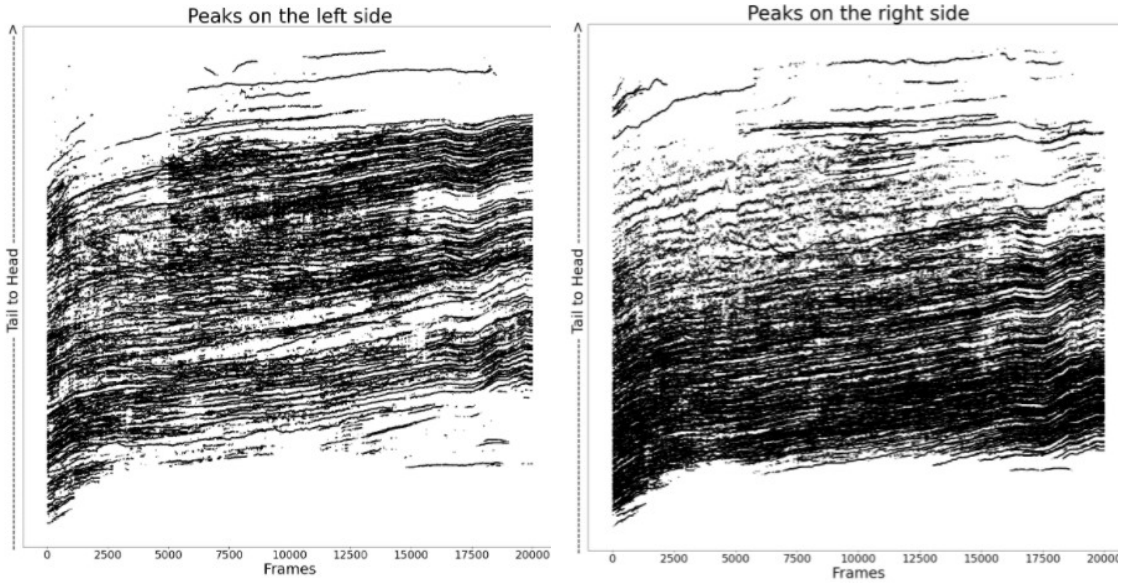
The three graphs show the number of objects for every movie, where each row corresponds to a different file. In each graph, the x-axis represents the frames and the y-axis the quantity.

6.1.2. Position of the intensity peaks over time

The position in y of the intensity peaks over time enables us to visualize where and when the embryo was most active and subsequently observe the synchronized activity. These graphs can be visualized in Figure 6.3, which has the same organization as Figure 6.1. The x-axis represents the frames, and the y-axis represents the peaks' position in y.

Left Side Right Side

Movie 1



Movie 2

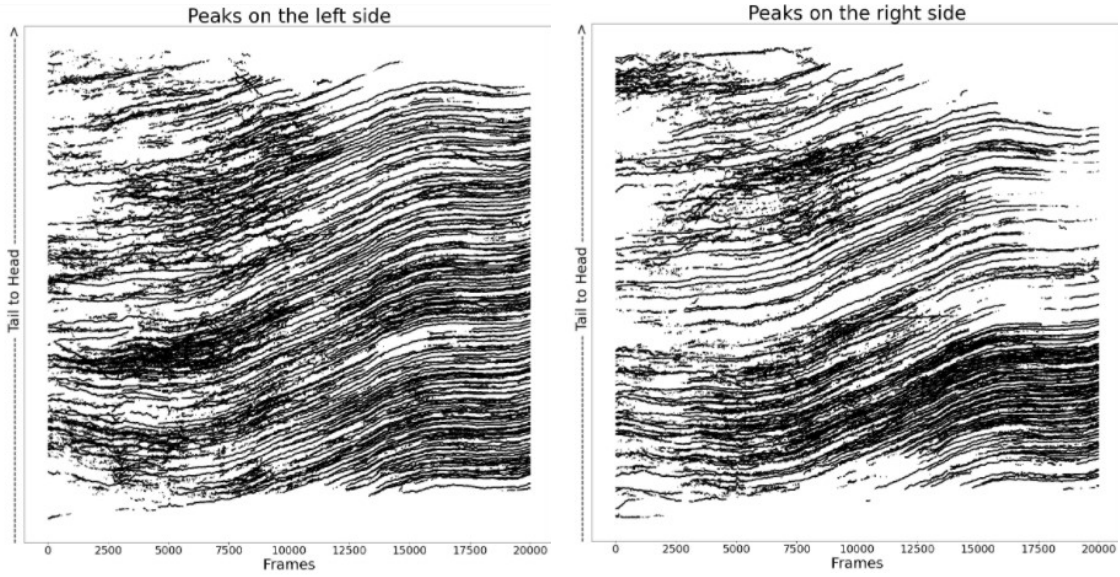
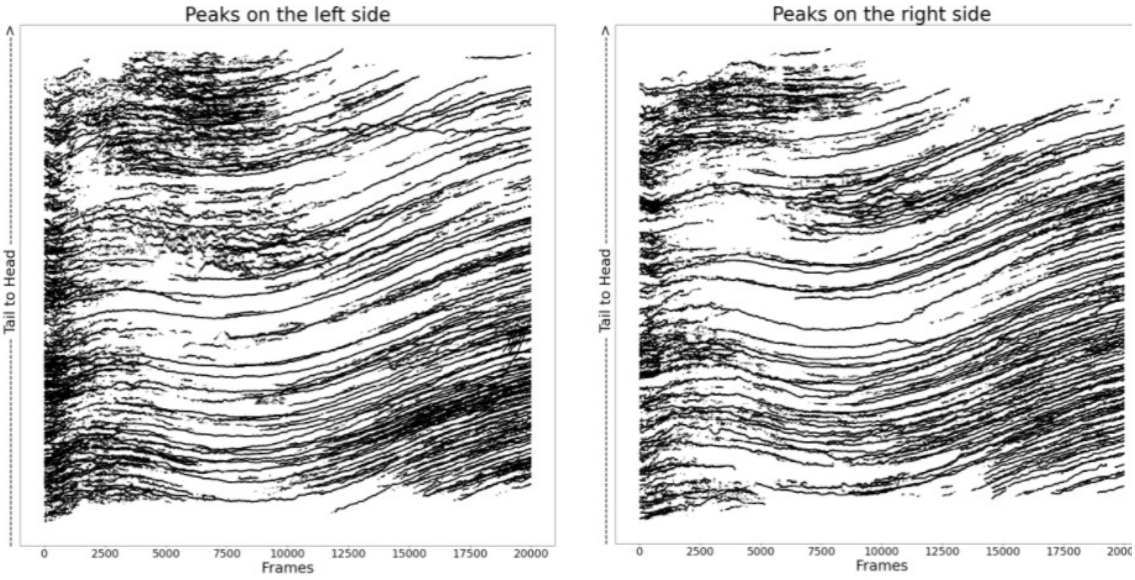


Figure 6.3: Intensity peaks' position in y over time

The six graphs represent the peaks Intensity peaks' position in y over time by movie and side. The rows represent different movies, and the columns correspond to the sides. The column on the left symbolizes the left side and the column on the right, the right side. The x-axis represents the frames, and the y-axis represents the peaks' position in y.

(Figure continues the next page)

Movie 3



6.2. NEURAL ACTIVITY AT A LATER STAGE OF EMBRYONIC DEVELOPMENT

6.2.1. Intensity over time of coordinated neurons

For the fourth movie's analysis, we wanted to visualize a representation of the intensities of a few coordinated neurons. With that aim, we studied, by direct observation, regions on the embryo which seemed to have neurons firing simultaneously, and the coordinates of those areas were extracted. Then, the analysis pipeline described previously was applied to extract the neurons' position and intensity over time. Finally, multiple sets of neurons were analyzed in order to select one that demonstrates the evolution of the intensity of coordinated neurons throughout time. Figure 6.4 shows a graph with ten neurons firing almost always at the same time.

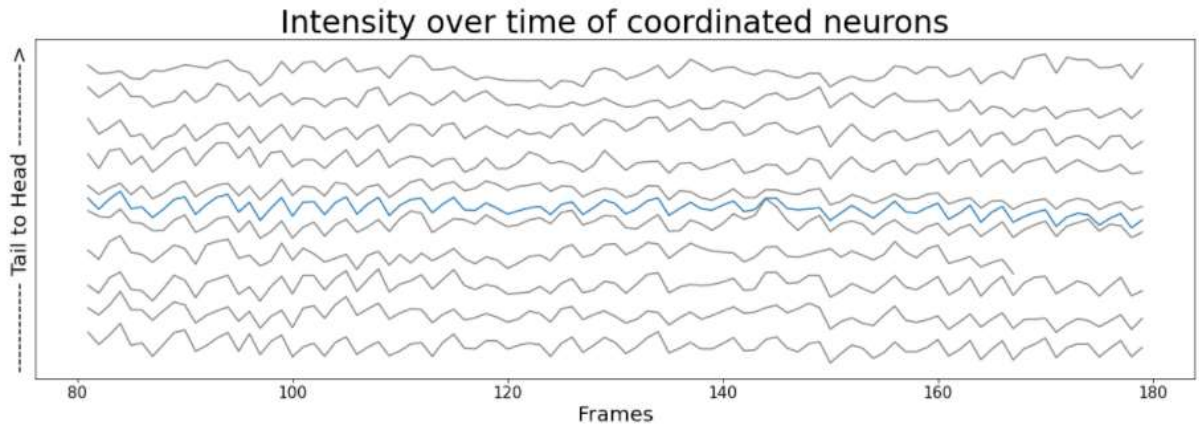


Figure 6.4: Intensity in time of coordinated neurons

The graph demonstrates ten lines (in gray), one for each neuron, with their peaks almost always in the same place in 100 frames. In addition, to a blue line that represents the sum of the intensities of the coordinated neurons. The x-axis represents the frames, and the y-axis is the intensity of the lines organized by their corresponding neuron's position in y.

Figure 6.5 demonstrates a representation of neurons firing in a small sequence of frames. Their intensities change throughout the sequence, where the images with brighter neurons correspond to intensity peaks of those neurons. The neurons shown were the same ones selected for Figure 6.4.

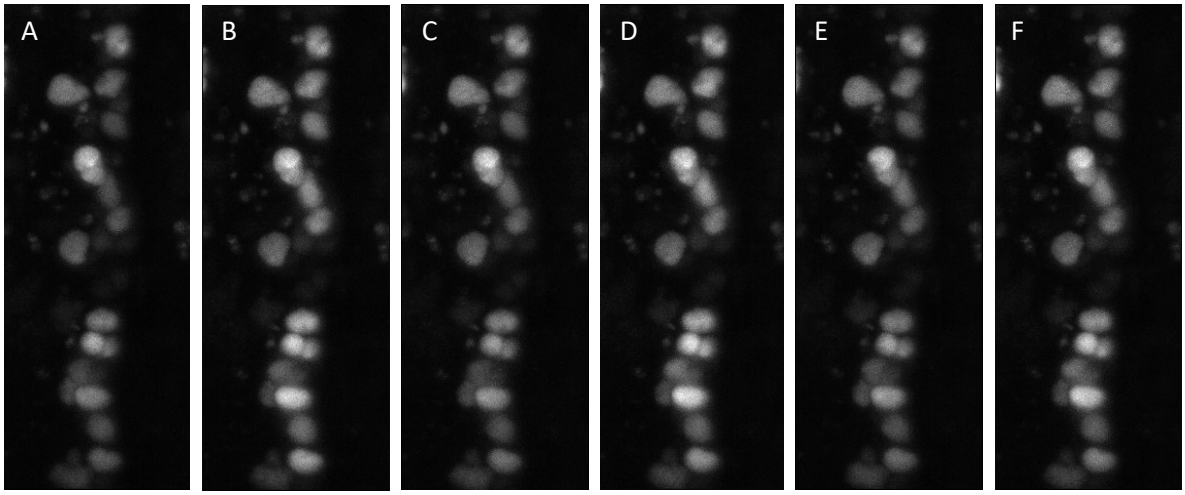


Figure 6.5: Sequence of frames showing the neurons firing

Each image corresponds to a frame. In frames A, C, and E, the intensities of the neurons are more deemed. In frames B, D, and F, the intensities of the neurons are brighter, corresponding to intensity peaks.

As a means of comparison, Figure 6.6 displays a graph similar to the one in Figure 6.4. However, in this graph, the intensities are representative of uncoordinated neurons from one of the three movies from the same time interval.

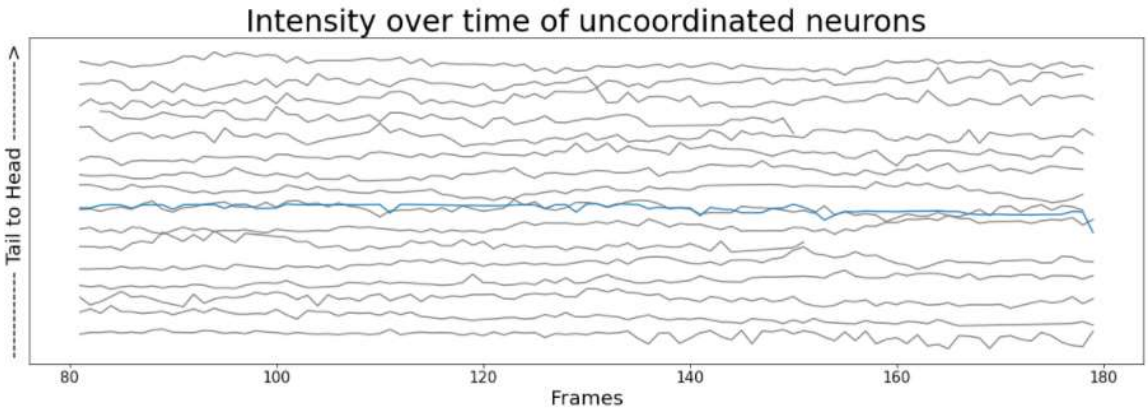


Figure 6.6: Intensity in time of uncoordinated neurons

The graph demonstrates the intensities over time of uncoordinated neurons in gray. In addition, to a blue trace that represents the sum of the intensities of the coordinated neurons. The x-axis represents the frames, and the y-axis is the intensity of the lines organized by their corresponding neuron's position in y.

7. DISCUSSION

This chapter presents a discussion about the results demonstrated in the previous chapter. In addition, it provides an analysis of the overall performance of the software.

7.1. NEURAL ACTIVITY OF THE EMBRYOS FROM THE SELECTED MOVIES

7.1.1. Frequency of the Intensity peaks

In Figure 6.1, we can observe that the different sides of the embryo have comparable behavior in each movie. The fact that some sides have a larger quantity of peaks than their opposite side is due to a high number of tracks in the formers. Furthermore, this difference could be a consequence of some error that might occur when dividing the embryo into two parts vertically. In other words, the line that makes this division could be slightly misaligned.

Movie 1: Both sides maintain more or less the same quantity of peaks. Overall, from the beginning to the end of the movie, there is a slight decrease of around 2000 peaks

Movie 2: Both sides have an initial increase in the number of peaks. The left side follows with a slight decrease but then increases again until the end. The right side, after the rise, maintains more or less the same quantity of peaks and then displays a slight decrease in the final frames.

Movie 3: Both sides have an exponential decrease of the number of peaks initially, followed by a slight increase until the end.

The decrease in the number of peaks in Figure 6.1 is correlated with a decline in the number of neurons in the same embryo, as can be observed in Figure 6.2. More specifically, the number of neurons in time in Movie 1 and 3 have the same behavior as the number of peaks on both sides of the respective movie. In Movie 2, the quantity of neurons is roughly steady in the end, but the number of peaks on the left increases and on the right decreases, so we could say that they also have the same behavior on average.

The decrease in the number of neurons could be related to the quality of the images. More precisely, it could be due to the general decrease of the intensity in the movies, as shown in Figure 4.4 in section 4.3, due to photophysical processes as photobleaching. In contrast, the increase in the number of neurons could be due to the development of the embryo. After all, as the embryo is developing, the number of cells is growing.

In conclusion, the frequency of the intensity peaks and the number of neurons over time suggest that we were entirely able to image and identify the active neurons since each intensity peak corresponds to the moment when a neuron is firing, implying it is active.

7.1.2. Intensity peaks' position in y over time

In Figure 6.3, we notice that there are almost parallel curves in all the movies that go upwards most of the time. In addition, the different sides of each embryo have the same behavior.

Movie 1 has a constant inclination upwards all the time

Movie 2 has short periods, one in the beginning and the other at the end, where the lines are steady. However, the rest of the time, the curves have a positive slope.

Movie 3 has a minimal movement downward in the first frames, followed by an inclination upwards.

The movements of the curves represent the migration of cells, which was possible to confirm by looking at the movies, where we can see the neurons on the embryos moving according to what is represented on the graphs.

In addition, by observing the number of the curves, we can understand where and when the neural activity was most active. In all three movies, we can see more quantity in the bottom part of the embryo. In particular, Movie 2 and 3 show a local increased neural activity in the that region. The concentration in the bottom part implies that the neurons are more active in the tail region of the embryo. Consequently, the position where the neurons are firing more suggests an association with the extension of the tail. Since the period when the tail elongates implies that the neurons need to be more active in that region.

There is a temporal shift among these movies due to the uncertainty in the fecundation time. However, it is possible to identify the specific temporal event corresponding to the tail extension in all three movies. In particular, in Movie 1, this event can be observed right at the beginning, but for Movie 2 and 3, it appears only in the second half of the experiment time, around the frame 12500 and 15000, respectively.

7.2. NEURAL ACTIVITY AT A LATER STAGE OF EMBRYONIC DEVELOPMENT

7.2.1. Intensity over time of coordinated neurons

Since we were not able to observe coordinated behavior in the very early stages of *Danionella's* development, we focused our attention on later stages. Here we could observe the coordinate firing of multiple neurons, as shown in Figure 6.4. In the graph, a subset of 10 neurons was followed over 100 frames, corresponding to 3.3 minutes, which are firing simultaneously, at a high temporal frequency, as can be seen by the sum of the intensities, which follows the same pattern as the intensities of the coordinated neurons.

As a comparison, Figure 6.6 shows the intensities of uncoordinated neurons in the same time interval from one of the three files analyzed previously. We selected a short time interval representative of the behavior of these neurons during the whole duration of the movie, and by observing the sum of the intensities, it can be seen that it does not follow any pattern, as opposed to the sum of the

intensities of the coordinated neurons. These preliminary results of imaging embryos at a later stage of embryonic development highlight the ability to extend our analysis pipeline to a more developed embryo.

7.3. PERFORMANCE OF THE SOFTWARE

The StarDist validation on section 5.2.1 showed that StarDist manages to identify and segment neurons in *Danionella translucida* embryos without the need to train a new model. Likewise, the graphs previously referred to in the previous sections also indicate that the tracking performed by TrackMate and the remaining steps of the analysis were significantly good. Since the quantity of neurons is proportional to the number of intensity peaks of active neurons, one could infer that there was no loss of information throughout the whole analysis.

To sum up, the software chosen were suitable and had a good performance.

8. CONCLUSIONS

An imaging method and an analysis pipeline based on deep learning were proposed to image *Danionella translucida* embryos and then investigate their neuronal activity as a way to achieve the aim of this dissertation. Accordingly, to the outcomes made throughout this project, several conclusions could be made:

- We were able to image several *Danionella translucida* embryos using light-sheet fluorescence microscopy.
- StarDist was effective in segmenting the neurons in the embryos imaged.
- TrackMate could track the segmented neurons. In particular, it could track the centroids of neurons from the tracking file that we successfully generated.
- We managed to identify and locate the neurons, specifying whether they belonged to the left or right side and whether they belonged to the bottom, center, or top region.
- We could visualize the onset neural activity and also the synchronized activity of a few neurons.
- We believe we could observe the consistent migration of cells and the incremental activity during the tail extension.

In conclusion, the methods chosen to image and analyze the data successfully studied the neural activity of Danionella translucida embryos imaged.

8.1. IMPLICATIONS BASED ON THIS PROJECT

This project proves it is possible to image developing *Danionella translucida* embryos during long periods of time using light-sheet fluorescence microscopy. Moreover, it is possible to perform deep learning-based analysis to identify, segment, and track the neurons in fluorescence images. Despite having studied only three full movies from embryos imaged at a similar embryonic stage, and a small subset from an embryo at a later developmental stage, it demonstrates great potential to perform a more in-depth study on the neural activity in *Danionella translucida* embryo. In addition, it shows a reliable procedure to analyze fluorescence microscopy images that can detect, track, and explore the signal in time.

8.2. LIMITATIONS

While the results demonstrate that the proposed procedure was an excellent approach to describe the neural activity of the *Danionella translucida* embryo, some limitations should be taken into consideration when making assumptions based on this study.

Due to the novelty of using *Danionella translucida* embryos in biological studies, it was challenging to know the correct time to image the embryos, more precisely when the transition from stochastic to coordinated neuronal activity happens. This problem led to only a few big movies with embryos in similar development periods that could be used for the analysis. Additionally, the neural activity was not fully synchronized in these movies.

Moreover, due to the limited computational power available and time constraints, it was impossible to image and analyze longer movies corresponding to longer imaging time or explore more movies.

Even with access to a computer with considerable capability, separating the files in quarters was necessary to perform the proper segmentation. Then, in the end, the labeled images from the different quarters were compiled into the correct order for the next steps. This implies that to use StarDist segmentation, it is necessary a computer with a RAM larger enough to handle the size of the big data files.

Consequently, three movies might not be enough to conclude about certain behaviors of the neurons.

8.3. FUTURE WORK

The recommendations for potential future work are highly associated with the limitations stated previously in this chapter.

Firstly, it is still an open question to observe the transition from stochastic to coordinated neural activity in *Danionella translucida* embryo. Even though we could not observe the coordination we expected in the stage of embryonic development that we examined, we were able to implement the necessary tools for that goal. Hence, future work could focus on imaging at a later stage and during more prolonged periods and, subsequently, with more computational power, analyze longer movies in order to describe the transition to synchronized neural activity.

Secondly, we were able to extract the neuron's position, and this information could be used in a detailed analysis. For example, instead of only comparing the sides of the embryo, it would be interesting to compare regions by bottom, center, and top. More specifically, tail, center, and head regions. We were indeed able to make this division. However, because of the time limit, we realized that it was enough to compare the sides of the embryo for this dissertation because it gave a general overview. In summary, all the information already extracted could be used for a more in-depth study of the neural activity in *Danionella translucida* embryos.

Moreover, the methods used for imaging and analysis based on deep learning could benefit not only neuroscientists, but the biological community in general, since they could be used to study signals from different types of cells that otherwise would be too complex to be analyzed. For example, it could be used to study the development of different organs in fish or the development of other model systems, like Drosophila.

Furthermore, this work could be extended to 3D detection and tracking since most organs and organisms depend on a correct 3D organization to function. So, limiting to 2D translates into a significant loss of information.

In conclusion, this project offers exceptional value for scientists that need to study long and complex biological processes that need the help of artificial intelligence algorithm to detect, track and extract meaningful information.

9. BIBLIOGRAPHY

Bernardello, M., Marsal, M., Gualda, E. J., & Loza-Alvarez, P. (2021). Light-sheet fluorescence microscopy for the in vivo study of microtubule dynamics in the zebrafish embryo. *Biomedical Optics Express*, *12*(10), 6237. https://doi.org/10.1364/BOE.438402

Chen, T. W., Wardill, T. J., Sun, Y., Pulver, S. R., Renninger, S. L., Baohan, A., Schreiter, E. R., Kerr, R. A., Orger, M. B., Jayaraman, V., Looger, L. L., Svoboda, K., & Kim, D. S. (2013). Ultrasensitive fluorescent proteins for imaging neuronal activity. *Nature*, *499*(7458), 295–300. https://doi.org/10.1038/NATURE12354

DeepCell. (n.d.). Retrieved November 19, 2021, from https://www.deepcell.org/

Girshick, R. (2015). *Fast R-CNN*. https://github.com/rbgirshick/

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). *Rich feature hierarchies for accurate object detection and semantic segmentation Tech report (v5)*. http://www.cs.berkeley.edu/˜rbg/rcnn.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org

Haykin, S., York, N., San, B., London, F., Sydney, T., Singapore, T., Mexico, M., Munich, C., Cape, P., Hong, T., & Montreal, K. (2009). *Neural Networks and Learning Machines Third Edition*.

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2018). Mask R-CNN. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *42*(2), 386–397. https://arxiv.org/abs/1703.06870v3

Henn, K., & Braunbeck, T. (2011). Dechorionation as a tool to improve the fish embryo toxicity test (FET) with the zebrafish (Danio rerio). *Comparative Biochemistry and Physiology. Toxicology & Pharmacology : CBP*, *153*(1), 91–98. https://doi.org/10.1016/J.CBPC.2010.09.003

Hill, A. J., Teraoka, H., Heideman, W., & Peterson, R. E. (2005). Zebrafish as a Model Vertebrate for Investigating Chemical Toxicity. *Toxicological Sciences*, *86*(1), 6–19. https://doi.org/10.1093/TOXSCI/KFI110

Howe, K., Clark, M. D., Torroja, C. F., Torrance, J., Berthelot, C., Muffato, M., Collins, J. E., Humphray, S., McLaren, K., Matthews, L., McLaren, S., Sealy, I., Caccamo, M., Churcher, C., Scott, C., Barrett, J. C., Koch, R., Rauch, G. J., White, S., … Stemple, D. L. (2013). The zebrafish reference genome sequence and its relationship to the human genome. *Nature*, *496*(7446), 498–503. https://doi.org/10.1038/nature12111

Hussain, M., Bird, J. J., & Faria, D. R. (2018). A Study on CNN Transfer Learning for Image Classification. *Advances in Intelligent Systems and Computing*, *840*, 191–202. https://doi.org/10.1007/978-3-319-97982-3_16

Icha, J., Weber, M., Waters, J. C., & Norden, C. (2017). Phototoxicity in live fluorescence microscopy, and how to avoid it. In *BioEssays* (Vol. 39, Issue 8). John Wiley and Sons Inc. https://doi.org/10.1002/bies.201700003

Image visualization and analysis. (n.d.). Retrieved November 18, 2021, from https://www.arivis.com/

Imaris for Cell Biologists - Imaris - Oxford Instruments. (n.d.). Retrieved November 18, 2021, from https://imaris.oxinst.com/products/imaris-for-cell-biologists

Jacquemet, G., Fazeli, E., Roy, N. H., Follain, G., Laine, R. F., von Chamier, L., Hänninen, P. E., Eriksson,

J. E., & Tinevez, J. Y. (2020). Automated cell tracking using StarDist and TrackMate. *F1000Research*, *9*, 1–9. https://doi.org/10.12688/f1000research.27019.1

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An introduction to Statistical Learning. In *Current medicinal chemistry* (Vol. 7, Issue 10). https://doi.org/10.1007/978-1-4614-7138-7

Jones, T. R., Kang, I. H., Wheeler, D. B., Lindquist, R. A., Papallo, A., Sabatini, D. M., Golland, P., & Carpenter, A. E. (2008). CellProfiler Analyst: Data exploration and analysis software for complex image-based screens. *BMC Bioinformatics*, *9*(1), 1–16. https://doi.org/10.1186/1471-2105-9-482/FIGURES/8

Judkewitz Lab — Imaging Microscopy Neuroscience Berlin. (n.d.). Retrieved November 26, 2021, from https://jlab.berlin/#about

Justusson, B. I. (1981). Median Filtering: Statistical Properties. *Two-Dimensional Digital Signal Processing II.*, 161–196. https://doi.org/10.1007/BFB0057597

Kaufmann, A., Mickoleit, M., Weber, M., & Huisken, J. (2012). Multilayer mounting enables long-term imaging of zebrafish development in a light sheet microscope. *Development*, *139*(17), 3242–3247. https://doi.org/10.1242/DEV.082586

Kimmel, C. B., Ballard, W. W., Kimmel, S. R., Ullmann, B., & Schilling, T. F. (1995). Stages of embryonic development of the zebrafish. *Developmental Dynamics : An Official Publication of the American Association of Anatomists*, *203*(3), 253–310. https://doi.org/10.1002/AJA.1002030302

Krogh, A., & Hertz, J. (1991). A Simple Weight Decay Can Improve Generalization. *Advances in Neural Information Processing Systems*, *4*.

Lecun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/NATURE14539

Li, Y., Qi, H., Dai, J., Ji, X., & Wei, Y. (2017). *Fully Convolutional Instance-aware Semantic Segmentation*.

Lichtman, J. W., & Conchello, J.-A. (2005). Fluorescence microscopy. *Nature Methods 2005 2:12*, *2*(12), 910–919. https://doi.org/10.1038/nmeth817

Meijering, E. (2020). A bird's-eye view of deep learning in bioimage analysis. *Computational and Structural Biotechnology Journal*, *18*, 2312–2325. https://doi.org/10.1016/J.CSBJ.2020.08.003

Mitchell, T. M. (1997). *Machine Learning*. 414.

Moen, E., Borba, E., Miller, G., Schwartz, M., Bannon, D., Koe, N., Camplisson, I., Kyme, D., Pavelchek, C., Price, T., Kudo, T., Pao, E., Graf, W., & van Valen, D. (2019). Accurate cell tracking and lineage construction in live-cell imaging experiments with deep learning. *BioRxiv*. https://doi.org/10.1101/803205

Mohri, M., Rostamizadeh, A., & Talwalkar, A. (2012). Foundations in Machine learning. In *The MIT Press*.

O'Shea, K., & Nash, R. (2015). *An Introduction to Convolutional Neural Networks*. https://arxiv.org/abs/1511.08458v2

Ren, S., He, K., Girshick, R., & Sun, J. (2016). *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. http://image-net.org/challenges/LSVRC/2015/results

Reynaud, E. G., Kržič, U., Greger, K., & Stelzer, E. H. K. (2010). Light sheet-based fluorescence microscopy: More dimensions, more photons, and less photodamage. *Http://Dx.Doi.Org/10.2976/1.2974980*, *2*(5), 266–275. https://doi.org/10.2976/1.2974980

Reynolds, A. H. (n.d.). *Convolutional Neural Networks (CNNs)*. Retrieved October 28, 2021, from https://anhreynolds.com/blogs/cnn.html

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9351*, 234–241. https://arxiv.org/abs/1505.04597v1

Rosebrock, A. (2016). *Intersection over Union (IoU) for object detection - PyImageSearch*. https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

Salinas, M., Rosas, J., Iborra, J., Manero, H., & Pascual, E. (1997). Comparison of manual and automated cell counts in EDTA preserved synovial fluids. Storage has little influence on the results. *Ann Rheum Dis*, *56*, 622–626.

Satsoura, D., Leber, B., Andrews, D. W., & Fradin, C. (2007). Circumvention of Fluorophore Photobleaching in Fluorescence Fluctuation Experiments: a Beam Scanning Approach. *Chemphyschem : A European Journal of Chemical Physics and Physical Chemistry*, *8*(6), 834. https://doi.org/10.1002/CPHC.200600589

Schindelin, J., Arganda-Carreras, I., Frise, E., Kaynig, V., Longair, M., Pietzsch, T., Preibisch, S., Rueden, C., Saalfeld, S., Schmid, B., Tinevez, J. Y., White, D. J., Hartenstein, V., Eliceiri, K., Tomancak, P., & Cardona, A. (2012). Fiji: An open-source platform for biological-image analysis. *Nature Methods*, *9*(7), 676–682. https://doi.org/10.1038/nmeth.2019

Schmidt, U., Weigert, M., Broaddus, C., & Myers, G. (2018). Cell Detection with Star-convex Polygons. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11071 LNCS*, 265–273. https://doi.org/10.1007/978-3-030-00934-2_30

Schott, B., Traub, M., Schlagenhauf, C., Takamiya, M., Antritter, T., Bartschat, A., Löffler, K., Blessing, D., Otte, J. C., Kobitski, A. Y., Nienhaus, G. U., Strähle, U., Mikut, R., & Stegmaier, J. (2018). EmbryoMiner: A new framework for interactive knowledge discovery in large-scale cell tracking data of developing embryos. *PLOS Computational Biology*, *14*(4), e1006128. https://doi.org/10.1371/JOURNAL.PCBI.1006128

Schulze, L., Henninger, J., Kadobianskyi, M., Chaigne, T., Faustino, A. I., Hakiy, N., Albadri, S., Schuelke, M., Maler, L., Del Bene, F., & Judkewitz, B. (2018). Transparent Danionella translucida as a genetically tractable vertebrate brain model. *Nature Methods*, *15*(11), 977–983. https://doi.org/10.1038/s41592-018-0144-6

scipy.signal.find_peaks — SciPy v1.7.1 Manual. (n.d.). Retrieved November 15, 2021, from https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.find_peaks.html

Sendhilnathan, N., Ipata, A., & Goldberg, M. E. (2021). Mid-lateral cerebellar complex spikes encode multiple independent reward-related signals during reinforcement learning. *Nature Communications*, *12*(1), 6475. https://doi.org/10.1038/S41467-021-26338-0

Singh, A. (2020). *Selecting the Right Bounding Box Using Non-Max Suppression (with implementation)*. https://www.analyticsvidhya.com/blog/2020/08/selecting-the-right-

bounding-box-using-non-max-suppression-with-implementation/

Skeletonize — skimage v0.19.0.dev0 docs. (n.d.). Retrieved November 14, 2021, from https://scikit-image.org/docs/dev/auto_examples/edges/plot_skeleton.html

Sommer, C., Straehle, C., Kothe, U., & Hamprecht, F. A. (2011). Ilastik: Interactive learning and segmentation toolkit. *Proceedings - International Symposium on Biomedical Imaging*, 230–233. https://doi.org/10.1109/ISBI.2011.5872394

Tinevez, J. Y., Perry, N., Schindelin, J., Hoopes, G. M., Reynolds, G. D., Laplantine, E., Bednarek, S. Y., Shorte, S. L., & Eliceiri, K. W. (2017). TrackMate: An open and extensible platform for single-particle tracking. *Methods*, *115*, 80–90. https://doi.org/10.1016/j.ymeth.2016.09.016

Van Der Walt, S., Schönberger, J. L., Nunez-Iglesias, J., Boulogne, F., Warner, J. D., Yager, N., Gouillart, E., & Yu, T. (2014). Scikit-image: Image processing in python. *PeerJ*, *2014*(1), e453. https://doi.org/10.7717/PEERJ.453/FIG-5

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Jarrod Millman, K., Mayorov, N., J Nelson, A. R., Jones, E., Kern, R., Larson, E., … van Mulbregt, P. (n.d.). SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods*. https://doi.org/10.1038/s41592-019-0686-2

Wan, Y., Wei, Z., Looger, L. L., Koyama, M., Druckmann, S., & Keller, P. J. (2019). Single-Cell Reconstruction of Emerging Population Activity in an Entire Developing Circuit. *Cell*, *179*(2), 355-372.e23. https://doi.org/10.1016/j.cell.2019.08.039

Weber, M., & Huisken, J. (2011). Light sheet microscopy for real-time developmental biology. *Current Opinion in Genetics & Development*, *21*(5), 566–572. https://doi.org/10.1016/J.GDE.2011.09.009

Wen, C., Miura, T., Voleti, V., Yamaguchi, K., Tsutsumi, M., Yamamoto, K., Otomo, K., Fujie, Y., Teramoto, T., Ishihara, T., Aoki, K., Nemoto, T., Hillman, E. M. C., & Kimura, K. D. (2021). 3deecelltracker, a deep learning-based pipeline for segmenting and tracking cells in 3d time lapse images. *ELife*, *10*. https://doi.org/10.7554/ELIFE.59187

Wolf, S., & Debrégeas, G. (2018). CHAPTER 1:Fast Volumetric Imaging Using Light-sheet Microscopy. Principles and Applications. *Comprehensive Series in Photochemical and Photobiological Sciences*, *18*, 1–24. https://doi.org/10.1039/9781788013284-00001

Zhang, X.-D. (2020). Machine Learning. *A Matrix Algebra Approach to Artificial Intelligence*, 223–440. https://doi.org/10.1007/978-981-15-2770-8_6