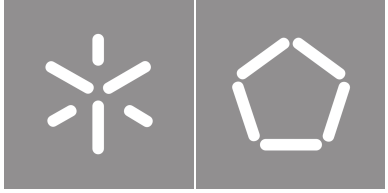
Universidade do Minho
Escola de Engenharia

Diogo Alexandre Rodrigues Lopes

Automatic Quantification of Microglial Cells from Brain Images

October, 2022



Universidade do Minho
Escola de Engenharia

Diogo Alexandre Rodrigues Lopes

Automatic Quantification of Microglial Cells from Brain Images

Master Thesis Dissertation
Master Degree in Informatics Engineering

Work developed under the supervision of:
Paulo Jorge Freitas de Oliveira Novais
Bruno Filipe Martins Fernandes

October, 2022

Acknowledgements

Countless times over the past year I have imagined writing these lines. For me, they mean closing a stage and reaching a goal. I always thought it would be the easiest task ahead. However, now that the time has come to write them, I see that I couldn't be more wrong. Including all the people who, over the years, have contributed to my formation as a Man and Engineer is a task doomed to failure. Still, I reserve a few words for those I couldn't help but mention.

First of all, I would like to thank my supervisors for the opportunity they gave me to carry out this work and for the teachings transmitted. To Professor Paulo Jorge Freitas de Oliveira Novais, I want to thank him for all the support, all openness he gave me and for allowing me to work in an area that fascinates me. To Professor Bruno Filipe Martins Fernandes, who over the last year was responsible for dealing with and following me more closely, a task that I admit I did not make easier, I cannot find words to thank him for everything he has done for me. That said, trying not to be unfair to him, I want to thank him for all the dedication, patience and availability he has always shown. Above all, I appreciate all the advice and guidance you gave me, advice that was not always related to my dissertation, but without a shadow of doubt helped me get here.

To my colleagues, with whom I had the fantastic opportunity to work and to cross paths in the last two years, I thank all the friendship and help. It was without a doubt a privilege to have learned with all of you and I am sure that great careers await you. To the others who shared moments with me in other years, a huge thank you for making me grow.

Last but not least, I thank my family, namely my parents and brother for all the love and unconditional support, dedication, concern, all our conversations and above all, all the countless sacrifices I know you had to do over these years. You made this all possible, allowing me to attend a higher education institution and to be finishing my dissertation. Thanks to you, I can now officially say that I have completed my Master's in Computer Engineering.

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the Universidade do Minho.

_____, _____

(Place) (Date)

_____

(Diogo Alexandre Rodrigues Lopes)

"If you really look closely, most overnight successes took a long time." (Steve Jobs)

Resumo

Quantificação Automática de Células Microgliais a partir de Neuroimagens

A microglia é um tipo de célula glial residente no sistema nervoso central e representa cerca de 10 a 15% da população de células cerebrais. Estas células não produzem impulsos elétricos, são responsáveis por processos fisiológicos e patológicos fundamentais, e representam a primeira linha de defesa dentro do sistema nervoso central. Assim, a quantificação destas células é fundamental num contexto clínico, pois permite uma melhor monitorização e planeamento de tratamentos para diversas patologias.

A contagem convencional de células envolve um conjunto específico de ferramentas e dispositivos desenvolvidos para esse fim. Este processo é demorado e impreciso devido a estar bastante dependente do operador. Atualmente, a maioria dos processos são feitos manualmente. No entanto, outras abordagens têm sido estudadas, com o intuito de melhorar o processo de contagem, para tornar o mesmo menos demorado, mais eficiente e reduzir o erro associado a fatores externos à contagem. Posto isto, o objetivo desta dissertação é o de estudar a melhor abordagem para automatizar a quantificação de células microgliais indo desde os métodos clássicos aos de *deep learning*. Combinado com as devidas técnicas de processamento e análise de imagem, a abordagem clássica mostra-se uma solução adequada. Contudo, nos últimos anos, abordagens baseadas em *deep learning* evidenciaram um desempenho promissor em várias tarefas de análise de imagens, como classificação, deteção e segmentação.

As abordagens desenvolvidas para automatizar o processo de quantificação foram testadas num conjunto de imagens construído em parceria com elementos da Escola de Medicina da Universidade do Minho. Quanto à abordagem da metodologia clássica, foi desenvolvido um protocolo dentro do ImageJ, que aliado com técnicas de processamento de imagem permitiu automatizar o processo de contagem. Com base em redes neuronais convolucionais, o problema de classificação referente a uma metodologia de *deep learning* obteve uma accuracy de 0.9021 e conseguiu classificar as 661 imagens em 5 minutos e 44 segundos. As duas abordagens, consideradas ótimas dentro de cada metodologia, são competitivas com os métodos do estado da arte, pois permitiram automatizar o processo, mostraram uma significativa melhoria na reprodutibilidade e eficiência.

Palavras-chave: Células Microgliais, Deep Learning, Processamento de Imagem, Quantificação Automática de Células, Segmentação de Imagem, Sistema Nervoso Central

Abstract

Automatic Quantification of Microglial Cells from Brain Images

Microglia are a type of glial cell residing in the central nervous system and represent about 10 to 15% of the brain cell population. These cells don't produce electrical impulses and are responsible for fundamental physiological and pathological processes, as they represent the first line of immune defence within the central nervous system. Thus, the quantification of these cells is essential in a clinical context, as it allows better monitoring and planning of treatments for different pathologies.

Conventional cell counting involves a specific set of tools and devices developed for this purpose. This process is time-consuming and imprecise due to being heavily dependent on the operator. Currently, most processes are performed manually. However, other approaches have been studied and developed to improve the counting process, making it less time-consuming, more efficient and reduce the error associated with factors external to the counting. That said, the objective of this dissertation is to study the best approach to automate the quantification of microglial cells, ranging from classical to deep learning methodologies. Combined with the appropriate image processing and analysis techniques, the classical approach proves to be an adequate solution. However, in recent years, approaches based on deep learning have shown promising performance in various image analysis tasks, such as classification, detection and segmentation.

The approaches developed to automate the quantification process were tested on a set of images built in partnership with researchers from the School of Medicine of the University of Minho. As for the classical methodology approach, a protocol was developed within ImageJ, which was combined with image processing techniques that allowed the automation of the counting process. Based on Convolutional Neural Networks, the classification problem referring to a deep learning methodology obtained an accuracy of 0.9021 and managed to classify the 661 images in 5 minutes and 44 seconds. The two approaches, considered optimal within each methodology, are competitive with the state-of-the-art methods, as they allowed for the automation of the quantification process, and showed a significant improvement in reproducibility, efficiency and reduced error associated with human factors.

Keywords: Automatic Quantification of Cells, Central Nervous System, Deep Learning, Image Processing, Image Segmentation, Microglial Cells

Contents

List of Figures

List of Tables

Glossary

cytomorphological Cell morphology. 7, 9

embryogenesis Sequential series of dynamic processes that include cell division and growth. 6

macrophages Type of white blood cell that helps eliminate foreign substances by engulfing foreign materials and initiating an immune response. Macrophages are the principal cells involved in chronic inflammation and usually become more prevalent at the site of injury only after days or weeks. 5, 7

neuroanatomist Neuroanatomists are an expert in the field of neuroanatomy. Neuroanatomy is the scientific study of the nervous system. 6

neuroglia Is one of several types of cells that function primarily to support neurons, and can be also called as glial cell. Neuroglia exceed the number of neurons in the nervous system. They exist in the nervous systems of invertebrates as well as vertebrates. 5, 6, 7

phagocytosis Process by which certain living cells called phagocytes ingest or engulf other cells or particles. 9, 10

phenotype Are all the observable characteristics of an organism that result from the interaction of its genotype with the environment. Examples of observable characteristics include behaviour, biochemical properties, colour, shape, and size. 5, 6, 9

Acronyms

2D Two-Dimensional 12, 22, 23, 24, 39, 58, 62
3D Three-Dimensional 24, 39

AI Artificial Intelligence 28, 29, 41, 42
ANNs Artificial Neural Networks 14, 30, 31, 32, 33, 36, 40, 42, 43

CNNs Convolutional Neural Networks 28, 29, 32, 33, 38, 39, 42, 44, 45, 69, 70, 75, 76, 78, 90, 98, 99, 100
CNS Central Nervous System 1, 5, 6, 7, 8, 9, 26
CSC Cervical Spinal Cord 47, 48

DCN Deep Cerebellar Nuclei 47, 48, 50, 51, 53, 62, 64, 65, 85, 86
DIP Digital Image Processing 11, 12, 13, 16, 39
DL Deep Learning 28, 29, 36, 41, 42, 100

FNNs Feedforward Neural Networks 31, 42

GANs Generative Adversarial Networks 36, 37, 43

IBA1 Ionized Calcium-Binding Adapter Molecule 1 8, 9, 10, 24, 37, 38, 46, 48, 63, 66
ITCN Image-based Tool for Counting Nuclei 57, 58, 60, 61, 62, 66, 67, 81, 83, 84, 85, 87, 88, 89, 90

ML Machine Learning 28, 29, 34, 39, 41, 42, 69

NN Neural Networks 29, 36, 37, 43, 69

PN Pontine Nuclei 47, 48

RNNs Recurrent Neural Networks 31, 32, 42

SVM Support Vector Machine 27

VAEs Variational Autoencoders 36, 37, 43

1 Introduction

The main objective of this study is to understand the best approach to automatically quantify microglial cells, focusing on classical and deep learning methodologies. This chapter aims to contextualize the reader for the goal of this work. The motivation behind this dissertation, along with the definition of the main objectives and contributions are described. Ultimately, this chapter also provides an overview of the document outline, summarising the main contents of each chapter.

1.1 Motivation

Microglial cells are one of the most important microorganisms in the Central Nervous System (CNS) and are about 10 to 15% of the brain cell population [1]. Microglia is a type of glial cell that doesn't produce electrical impulses. They are responsible for fundamental physiological and pathological processes, representing the first line of immune defence within the CNS [2]. Upon detection of any sign of brain damage or nervous system dysfunction, these cells undergo an activation process. In the activation process, these cells migrate to the site of injury [3]. Given that importance, the quantification of these cells is fundamental in a clinical context, as it allows for better monitoring and planning of treatments for different diseases.

Therefore, cell count is an indispensable procedure routine that, in some cases, helps in the detection of a particular disease. Most of the cell quantification processes are performed manually. This quantification task requires time and is a tedious process. The final results may vary considerably between users, even when they are very experienced. On the other hand, automatic approaches have been studied and proposed over the last few years. The automated counting process has proven to be faster and more effective. This approach gained importance in the medical context, as evidenced by the inclusion of several automatic quantification and segmentation systems in a clinical environment.

To overcome the challenging task of the automatic quantification of microglial cells, some available alternatives are derived from the so-called more classic and deep learning-based approaches. Regarding the classical approach, an image that contains scattered cells in a layer and a known area, software solutions and assistants for automatic cell counting are applicable and make the quantification process easier [4]. Deep learning is a branch of machine learning and has been successfully applied to hard

tasks. In this sense, it stands as a good choice for automatic quantification of microglial cells, essentially due to its robustness, even in complicated domains with a lot of data variance. Deep learning methods learn from data instead of rule-based programming. Hand in hand with the different approaches we have indispensable image processing and analysis techniques. Image processing techniques convert an image into a digital form, and then further processing is allowed in those images [5]. The image analysis technique emulates human vision [6], including learning and the ability to make a decision based on input. Morphological operators are also a key subject in image analysis.

So, to solve the problem of automatic quantification of microglial cells, a study and conception of solutions to automatically quantify these cells, within the classical and deep learning approaches, will be carried out. In this context, this dissertation presents an introduction to classic and deep learning approaches for automatic cell counting. The state-of-the-art techniques are a good reference for it since they were meant for this challenge. In the same way, those concepts can be adapted to less complex but demanding tasks that require a high level of accuracy.

1.2 Objectives

The potential of automatic cell counting methods is, without a doubt, very vast. From the outset, this procedure can provide the medical context with a greater capacity to analyze large sets of images, substantially reducing costs and time lost with these tasks. Then, the methodologies for automatic cell counting open doors for the creation of new systems, which can reformulate the diagnosis and treatment of various diseases.

The problem is that nowadays most cell counting processes are manual. Such processes are time-consuming, tedious, and imprecise, being heavily dependent on the operator. Considering the automatic quantification of cells problem, this dissertation aims to develop means to quantify, i.e., count, the number of microglial cells from brain images. As stated, the expectation is that we will be able to conceive a method that, with acceptable performance, can automatically quantify microglial cells.

The first objective is to study and evaluate the different approaches to quantify the number of microglial cells. The second objective is to investigate from the classical methodology proposed solutions, which is the most suitable one to solve the problem, and successfully apply it to the automatic quantification of microglial cells from brain images. At this stage, the software and assistants for cell counting will emerge allied with image processing and analysis techniques. The third objective is to conceive and evaluate a deep learning-based approach to automate cell counting. Such approaches have already evidenced promising performance in various image analysis tasks, such as classification, detection, and segmentation, as shown in future lines. The last objective is to compare the results between the classic and deep learning approaches to fundament which one is the most suitable solution to automatic quantify microglial cells.

1.3 Contributions

This dissertation gave rise to two scientific publications. The first, entitled "Reviewing Computational Approaches for the Automatic Quantification of Cells from Brain Images" by Diogo Lopes, Ana Bela Campos, Patrícia Maciel, Paulo Novais and Bruno Fernandes is a review of different computational approaches for the automatic quantification of cells. In this scientific publication, the classic and deep learning-based approaches were studied, helping to fundament a strategy to solve the problem raised in this dissertation. This scientific publication was submitted for publication in the 17th International Conference on Practical Applications of Computational Biology & Bioinformatics.

The second, entitled "A Benchmark between Classic and Deep Learning Methods for the Automatic Quantification of Microglial Cells", is under preparation for submission in the Computer Methods and Programs in Biomedicine journal, a Q1 journal in the domains of computer science applications and health informatics. This scientific publication consists of a compilation of all the work developed, within the classical and deep learning methodologies, for the presented problem. This paper presents the results, the corresponding discussion, and all the main conclusions drawn from the work developed in this dissertation.

1.4 Dissertation Structure

This dissertation is organized into seven chapters. These are oriented to present the reader with all the relevant information collected during the literature review, carried out in the state-of-the-art on the main topics underlying this dissertation. The entire process of the quantification of the microglial cells is given.

Chapter 1 is intended to guide the reader to the subject of this work, presenting the context and motivation for carrying out this study. Following that, and once delivered the problem that led to this study, the main objectives and expected results of this dissertation are presented. The remainder of this section provides a general overview of the document outline.

We proceed by delivering a medical perspective on microglial cells in Chapter 2. We start with a general overview of microglial cells, namely their origin and morphology. The reader is then presented with how these cells can be identified emphasizing cell markers since they will always be at the centre of our study.

Chapter 3 consists of a literature review on the different approaches for the automatic cell quantification of microglial cells from brain images, especially the classic and deep learning approaches. Concepts of image processing and analysis techniques are presented and distinguished with some examples of the most common methods. The emphasis given to these techniques is related to the fact that they are key in digital image processing.

The information related to the data used to study and conceive a solution to the problem is provided in Chapter 4. Firstly, the generic cell samples used to understand the best approach to take with ImageJ, deep learning algorithms and their respective limitations, are presented. Next, the set of neuroimages

used to build the dataset is also described. All the preparatory work carried out with them is presented to the reader in this chapter.

Chapter 5 presents to the reader all the implementations developed to conceive a solution based on some of the statements and conclusions drawn previously from the studied methodologies and approaches. It also presents the experimental environments where all the experiences were conducted.

The results obtained from the study are given in Chapter 6. In this chapter, the reader can understand the entire analysis and discussion of the reached results.

Finally, Chapter 7 closes this document by presenting the main conclusions and perspectives of future lines of research related to the study developed to this point.

2

Microglial Cells - Medical Prespective

This chapter presents the reader with a medical perspective on glial cells, focusing on microglial cells. These macrophages are a type of neuronal support cell present in the central nervous system of invertebrates and vertebrates. Hence, these cells function primarily as immune cells they gather a big research interest. To better access and comprehend their appearance in neuroimaging, a study related to shapes that this organism adopts and cell markers are presented, as both have a tremendous role in the identification for prior quantification. Image analysis of glial cells is the main application of the study carried out with this dissertation, specifically microglial cells quantification from brain images. So, this chapter presents theoretical foundations and gives a technical background related to these organisms, as they play an important role in the implementation of an automatic quantification system.

2.1 General Overview

Microglia are a type of neuronal cell (neuroglia) located throughout the brain's spinal cord. They represent around 10 to 15% of the brain cell population [1]. As stated, microglial cells are the resident macrophages in the CNS that don't produce electrical impulses. Glial cells are responsible for fundamental physiological and pathological processes. They represent the first line of immune defence within the CNS because they constantly monitor the CNS microenvironment for plaques, damaged or unnecessary neurons and synapses, and infectious agents [2]. According to a study published in the Journal of Clinical Investigation, microglia are also responsible for other cerebral processes, such as the regulation of synaptic architecture and neurogenesis. They are also fundamental for sustaining normal brain functions under healthy conditions [7]. These cells migrate into all CNS regions and acquire a specific ramified morphological phenotype, which is a physical observable property like the one we can see in the image below, termed "resting microglia" [3]. This dissertation's main goal is to quantify this morphological phenotype from brain images.

Microglial cells are considered the most vulnerable sensors to brain pathologies. Surveillant microglia can be activated, adapt and respond according to a specific stimulus. The detection of any sign of brain damage or nervous system dysfunction may be triggered by neurological degenerative disorders such as

Alzheimer's and Parkinson's disease or by infections. These cells undergo an activation process that converts them into the "activated microglial cell". When activated, microglial cells can move to the site of injury. Recent studies suggest that these cells can help delay the progression of diseases in the brain. Figure 1 shows an example of microglia cells from a specific part of a brain image used in this study.

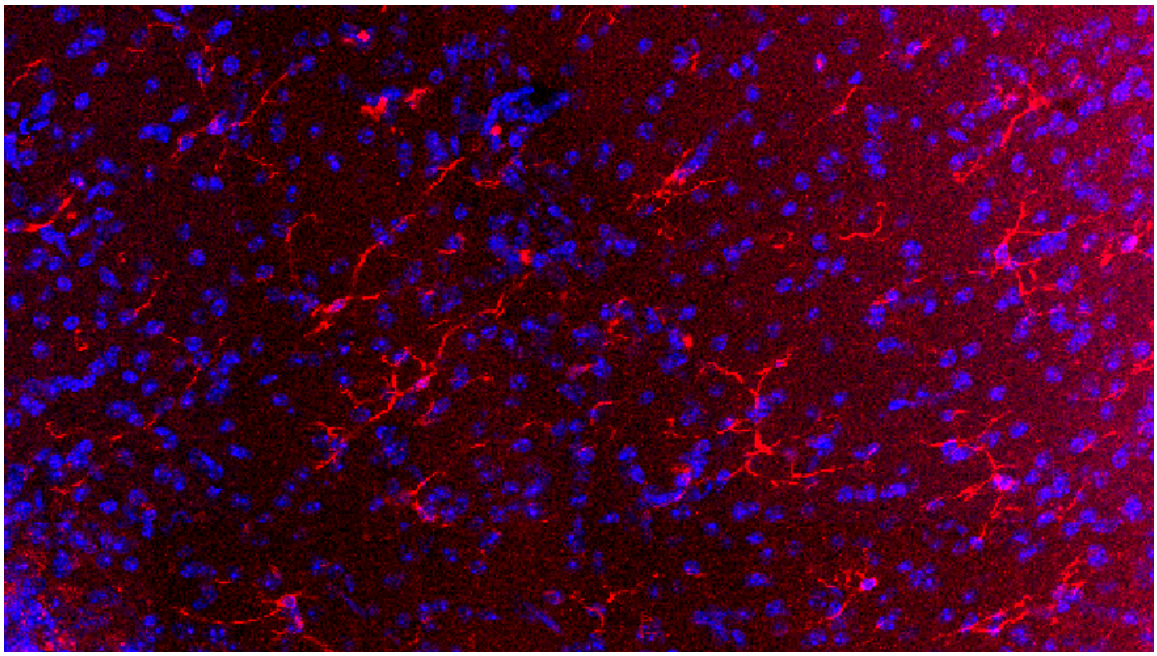


Figure 1: Microglial Cells.

2.2 Origin and Development of Microglial Cells

Microglia were first discovered by a Spanish neuroanatomist Pio del Rio-Hortega, between 1919 and 1921, through a study involving histological staining with silver carbonate. This neuroanatomist defended that microglia are distinct from other types of neuroglia because they derive from embryonic mesoderm, which gives rise to the blood and immune system cells [3].

The origin of microglial cells has been a theme long time debated. The evolutionary origins of these cells remain largely unexplored. However, today it is already consensual that these cells are derived from progenitors from the bone marrow and migrate into the CNS during the process called embryogenesis [8]. Once they reach the brain, glial cells propagate and disperse, in a non-heterogeneous manner, throughout the CNS. After invading the brain, glial cells transform into a ramified phenotype. It's important to highlight that in a healthy CNS, including the brain and spinal cord microglial cells have a ramified morphology. This specific morphology will help the identification and later quantification of cells. In a healthy brain, microglial cells exist as a stable population.

2.3 Identification and Morphology of Microglial Cells

Neuroimaging offers the non-invasively possibility to assess the brain. These images are a rich source of data that allows the medical community and researchers to extract measurements, locate and study lesions to prepare certain treatments. For these purposes, and remembering that microglial cells are key for fundamental physiological and pathological processes, their identification throughout the brain images is essential. Additionally, these images contain information that allows us to visualize the cell behaviour through their morphology and the specific shapes acquired.

Regarding the information presented above, this section presents some theoretical foundations related to microglial cell identification and morphology, as they are a key subject and will help in the decision-making of the system based on a more classic and deep learning approach to quantify these cells.

2.3.1 Identification

The identification of glial cells beyond cytomorphological criteria has been facilitated by the development of staining procedures that take advantage of the unique expression of certain molecules in certain cell types. Regarding the purpose of the dissertation, this will be helpful when trying to identify and count the number of cells using automatic cell counting related to more classic methods, by setting contrast and colour variations in pixel values. In section 3.3, we can find more detailed information on how these specific methods and techniques should be applied based on the literature collected, referring to the application of this type of method in similar situations.

Microglia can be seen in some sections of human brain tissue and are well distinguished from other brain cells CNS [3]. Regarding the purpose of this study, deep learning methods are expected to easily identify these cells when trained to recognize and distinguish them. The different approaches that deep learning methods can take, namely learning algorithms applied to similar problems, clustering, deep clustering, generative networks, object detection and image segmentation, are described in detail in section 3.4, based on previous applications of automatic cell counting.

2.3.2 Morphology

Regarding these cells' morphology, it is consensual that they are the smallest of all neuroglia. They are typically oval-shaped, and projecting out from their bodies exist slender elongated processes that allow these cells to move. These cells undergo a variety of structural changes based on their location and immune system needs. The ability to transform themselves distinguishes them from other macrophages. This helps to defend the CNS on extremely short notice without causing any immunological damage.

According to the information presented above, microglial cells can go from a resting state to an active state. Microglia reduce the complexity of their shape when they shorten or retract their branches. The shapes that these cells acquire are "Ramified or Hyper-Ramified Shape" and "Reactive Shape", as we can

see in Figure 2. The microglial cells seen in vitro do not usually have a branched structure compared to the microglial cells typically seen in the normal CNS [3].

Ramified and Hyper-Ramified Shape Microglial cells assume this particular shape throughout the CNS, specifically throughout the spinal cord and brain. This form is composed of long branches and a small cellular body. The cell body of this ramified shape remains stable while its branches, which are very sensitive to small changes, are constantly moving and supervising the surrounding areas. Microglia in this state can search and identify various immune threats. Although this is considered to be a "resting state", microglia that remain ramified are extremely active and can be quickly transformed into the activated form to respond to any injury or threat [9].

Reactive Shape Most of the time, when we refer to microglia, we use the term "activated", although some scientific studies assert that it is no longer correct. We should use the terminology "reactive" microglia. The term used is misleading as it indicates a polarization of cellular reactivity. The Ionized Calcium-Binding Adapter Molecule 1 (IBA1) marker, detailed in subsection 2.4, is upregulated in reactive microglia and is often used to visualize these cells. This is the marker used to help the visualization of the cells in this dissertation through the classic and deep learning methods [7]. When reactivated, microglial cells became phagocytic activated and non-phagocytic. When glial cells became phagocytic activated are the best line of an immune-responsive form that microglia can be. Non-phagocytic cells are glial cells that are moving from their ramified form to their fully active phagocytic form [10].

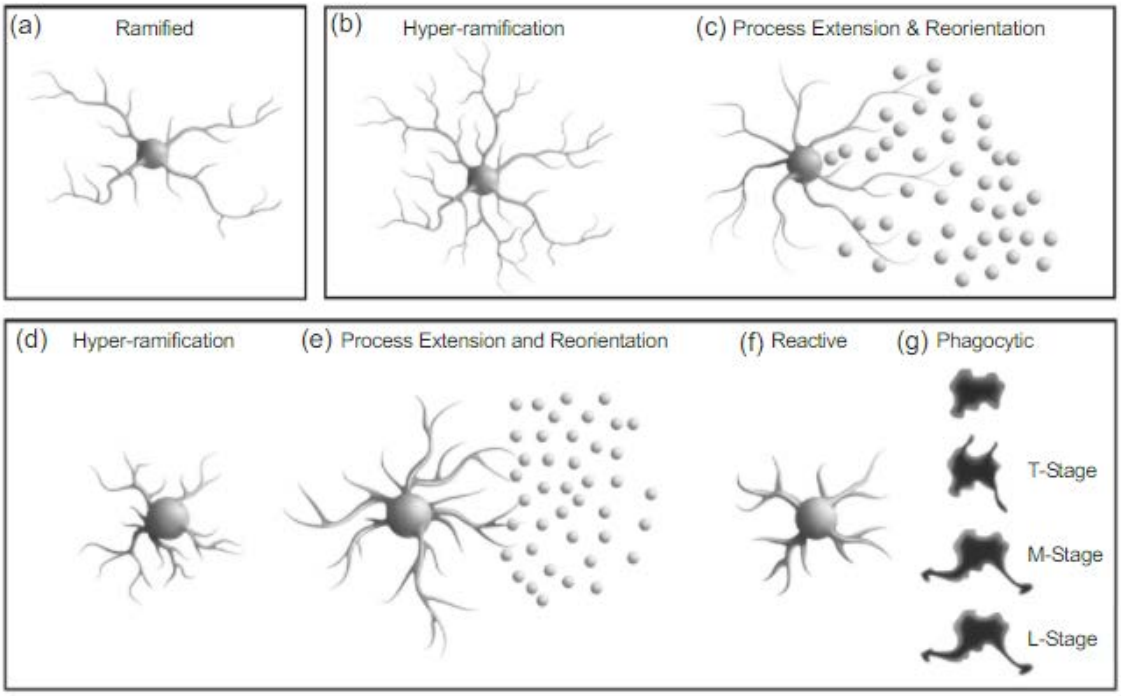


Figure 2: Representation of Microglial Cells Morphology. Source [11].

2.4 Markers of Microglial Cells

To recap, microglia are resident macrophages in the CNS that play an immune defence role. As stated in subsection 2.3.2, they exist assuming a ramified shape and are actively investigating the surrounding areas for injuries or infections. Being static but always looking for threats, they can be quickly "activated" in response to environmental changes. Once a threat is detected, the microglia undergo some morphological changes. Activated microglia are divided into essentially two states, M1 and M2, based on their morphology. M1 microglia are the first cells to respond to injury, while M2 microglia, also known as the anti-inflammatory microglia, can promote regression of neuroinflammation and stimulate tissue repair [12].

Regarding the microglia marker, there are three different state markers the Steady-State, M1 and M2 Markers. When choosing microglia markers for cell visualization, their location is essential. These markers help us to identify the cell more easily. There are numerous markers for microglial cells, but it is important to point out, that in this dissertation, the samples used to study a way to automatically quantify the number of cells contain the positive and non-positive IBA1 cell marker on them. IBA1 or AIF1 is a cytoplasmic protein, related to microglia motility and phagocytosis and is associated with microglial cell activation. IBA1 is the protein of a ramified microglia [13].

2.5 Summary

Microglial cells are a type of support CNS neuronal cell that don't produce electrical impulses. These cells reside in the healthy CNS parenchyma and are responsible for fundamental physiological and pathological processes. Glial cells include oligodendrocytes, astrocytes, ependymal cells, and microglia, with this last one representing 10 to 15% of the brain cell population. Microglia function primarily as immune cells because they constantly monitor the CNS microenvironment, being able to detect extracellular changes. Located throughout the brain spinal cord, and in addition to their immune functions, these cells are important in other brain processes such as the regulation of synaptic architecture. These cells migrate into all central nervous system regions and acquire a specific ramified morphological phenotype.

Microglia were first discovered between 1919 and 1921 through a study involving histological staining with silver carbonate. However, the origin of microglial cells has been a long-debated topic. Despite this, it is already consensual that these cells are derived from progenitors and migrate into the CNS. When they reach the brain, these cells propagate and disperse in a non-heterogeneous manner throughout the CNS, transforming themselves into a ramified phenotype. They can go from a resting state to an active state. This transition is accompanied by morphological changes. Microglia reduce the complexity of their shape when they shorten or retract their branches.

Regarding the purpose of this dissertation, which is the automatic quantification of these cells from brain images, the identification of microglial cells is crucial. The identification of these cells beyond cytomorphological criteria has been facilitated by the development of staining procedures. These procedures

take advantage of the unique expression of molecules in cell types, helping to identify and count them. By setting contrast and colour variations in pixel values, we can count cells when we face a more classic approach. Microglial cells are easily distinguished from other brain cells, which will be readily identified by deep learning methods when trained to recognize and distinguish these cells. Another key point to help in the identification of these cells for posterior counting are the cell markers.

About the markers, there are three different state markers, the "Steady-State" and "M1 and M2 Markers". All the samples used in this study have the IBA1 positive and non-positive marker on them. IBA1 is a marker related to microglia motility and phagocytosis, as well as microglial cell activation.

3

State of the Art

To quantify microglial cells present in the central nervous system, its identification through neuroimaging is indispensable. Roughly, we can do this quantification in two different ways. The manual quantification task presents itself as more time-consuming and error-prone even for the most experienced specialists. As an alternative, automatic quantification systems using classic and deep learning methodologies have shown robust results and the potential to reduce the human error associated with manual quantification. In addition, they have proven to be faster and more effective. They gained importance in the medical context, as evidenced by the inclusion of several automatic quantification and segmentation systems.

This chapter introduces the reader to relevant theoretical foundations and gives a technical background related to the scope of this dissertation. Image processing and analysis techniques are required to implement a solution to automatically quantify microglial cells. Therefore, a description of the global aspects of image processing and analysis is presented alongside with the processing techniques applied to enhance some characteristics in brain images and the morphological operators of image analysis. Next follows a brief contextualization of the automatic quantification process. Emphasis is on ImageJ, as it will be the software of choice and analysis of similar work developed regarding this classical approach. This chapter also provides information related to deep learning methodology and its respective algorithms that have been applied successfully to image classification and in this way automate the quantification of cells. As deep learning is a machine learning technique, emphasis is on learning paradigms, object detection and recognition.

3.1 Image Processing Techniques

Image processing is the methodology that allows converting an image to a digital aspect. Then, enabling us to perform actions on it, to get an enhanced image or extract quantitative and qualitative information from images using appropriate techniques. Today, image processing is spreading throughout various fields. Therefore, distributed into several groups emerge "Visualization", "Image Retrieval", and "Digital Image Processing (DIP)". Bering in mind the goal of this dissertation, the last-mentioned element of the various image processing groups is the one that needs to be pointed out and will be detailed later.

It is worth explaining that an image is a set of distributed data in an array. The positions are defined by elements from the Cartesian plane [14]. Therefore, data values are described as a Two-Dimensional (2D) function, $f(x, y)$, where x and y, are coordinates of the Cartesian plane, and assume integer values from 0 to $N-1$ and $M-1$, respectively, in images with size $N \times M$. Figure 3 illustrates and substantiates the presented information.

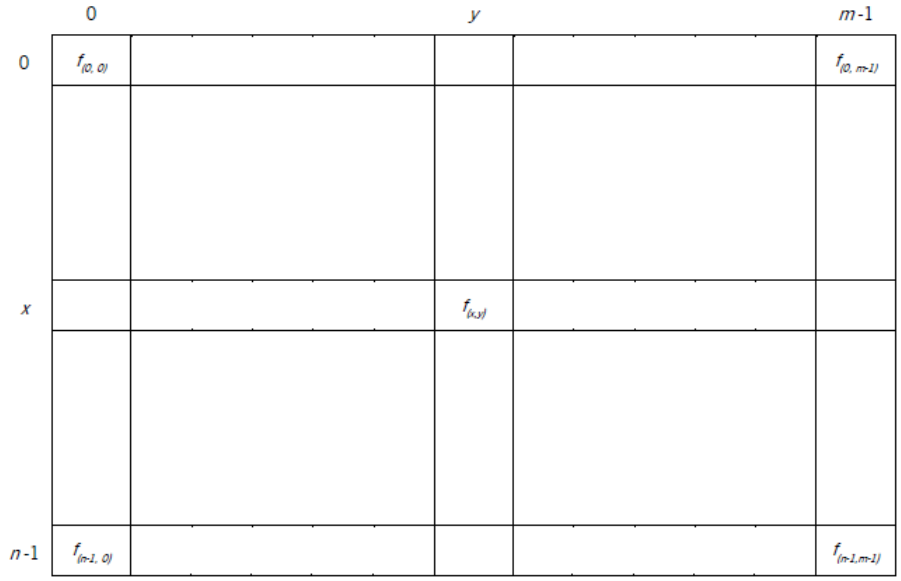


Figure 3: Mathematical Notation for a Digital Image. Adapted From [14].

Sometimes, the images may be squares, making M equal to N. Another repeatedly used terminology is the given spaces generated by the intersection of rows and columns of a matrix. That said, i and j represent the row and column numbers, respectively, with the same range of integer values. Each element of this matrix or grid is named by picture element, image element, or pixel, being pixel the best-known denomination [14]. Data values, f, is a discrete representation of the intensity or amount of visible light reflected by an object. Considering our informatics background, we know that an image is visualized by the computer as an array of integers. The application of algorithms for array manipulation is a common practice. So, image processing needs several techniques available with the assistance of computer programs. However, it's possible to extract qualitative and quantitative information from images using mathematical theorems, despite being a complicated and demanding task.

DIP is the technique of processing images performed by computers, eliminating the possibility of extracting information using mathematical theorems. First, the images are converted into a digital form, acquiring a computerized structure, and then further preparation/processing is done on those images. To obtain this computerized structure, image processing uses various techniques such as correction, formatting of the data, and enhanced procedure to create images with better quality [5]. It is important to point out that DIP allows the usage of complex algorithms for more sophisticated or simpler tasks. DIP is a tangible application of classification, feature extraction and pattern recognition. When we talk about

DIP, there are several techniques associated with it, more precisely seven different techniques. However, we will only focus on four of them, since they are related and are a key element in the automatic cell quantification process using classical and deep learning methodologies. These techniques are "Image Segmentation", "Classification", "Image Restoration"and "Image Enhancement".

3.1.1 Image Segmentation

Image Segmentation allows the partitioning of an image into several regions, several subparts, or even splitting it into pixels, according to the requirements intended by the user. This approach is often used for the analysis of substances, borders and other records relevant to processing [15]. The outcome of image segmentation is a set of sections that cover the total image. The main goal of segmentation is to simplify a raw image in such a manner that makes it easier to evaluate a complete picture. The segmentation of images is performed to compress images and facilitate the recognition of objects and editing purposes. Bearing in mind this dissertation, the part of the recognition of objects associated with the image segmentation will be extremely helpful when applying a more classic approach for automatic cell counting, as described in section 3.3. For that, thresholding the image is also indispensable. Segmentation allocates labels to each pixel so that pixels have similar labels and can share features [16]. This helps the identification and subsequent quantification of cells. There are several image segmentation techniques, as we can see in Figure 4. These techniques are detailed further below.

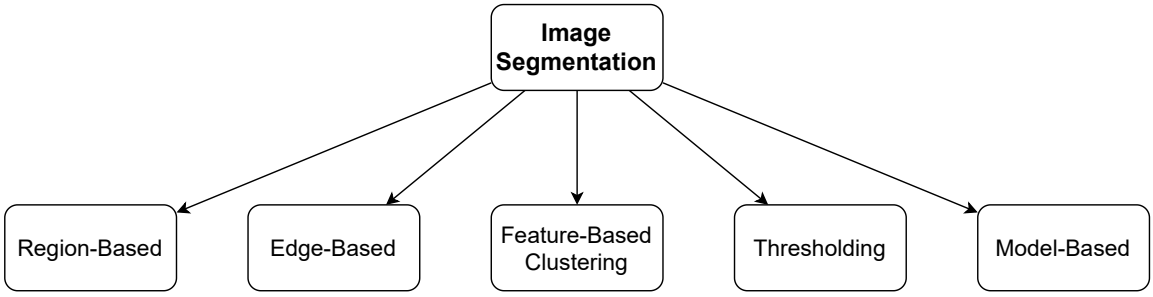


Figure 4: Image Segmentation Techniques.

Region-Based This technique groups the objects used for segmentation. The region-based method, also known as similarity-based segmentation, requires that certain regions must be together with each other so that the segmentation can take place. The borders of an image are recognized to perform segmentation. The area that is detected for segmentation should be closed, and the thresholding technique is bound with region-based segmentation [17]. Every step of this technique requires at least one pixel for processing purposes. The colour and texture of the image are altered, and a vector is created from the edge flow. Then, further processing is applied to those edges [18].

Edge-Based Another technique used in image segmentation is the edge-detection method also known as edge-based. This technique is formulated as a binary classification problem at the pixel level to identify individual pixels [17]. To recognize pixel values, edges are drawn, and then these edges are compared with other pixels. The basic procedure of this technique starts with extracting information about the edges. Then labelling is done for pixels. The segmentation is performed by the edges, and they must be far from each other. The linking is performed to fill the gap between the edges [18].

Feature-Based Clustering Feature-based clustering, commonly known as clustering, is another option to perform image segmentation. With clustering, an image is changed into a histogram. Following that, the clustering technique itself can be applied to the image. Pixels of the coloured images are clustered for segmentation using an unsupervised technique (Fuzzy C-Means Clustering). The stated procedure is applied to ordinary images, but if there is detected some noise, the result will be image fragmentation [19].

Threshold Thresholding is considered the easiest method used for image segmentation. This approach changes a grayscale image into a binary image wherever the two points are allocated to pixels. These two points are located below and on the upper side of the definite threshold value. The threshold value is obtained from the histogram of the original image and is calculated by the detection of edges which implies that this value is only correct if the detection of the edges is accurate. Segmentation perform via thresholding has lesser calculations in comparison to other methods [20].

Model-Based This technique is based on Markov arbitrary field. For colour segmentation, inbuilt region constraints are implicit. To characterize the exactness of the edges MRF is joined with edge detection. This method contains the relations amongst colour components [21].

3.1.2 Classification

Classification is the technique used to extract data and pixels from images. To perform classification, the minimum requirement is to have as many samples of similar objects as possible. An appropriate classification scheme and an adequate amount of training samples are the basics for effective classification [22]. There are various classification approaches such as Artificial Neural Networks (ANNs) and Fuzzy Logic. The classification technique is either supervised or unsupervised. In supervised classification, spectral signatures are obtained from training samples and are used to classify an image. After that, from the given training pieces, a signature file is assembled. With the help of classification tools, the image is then classified. In unsupervised classification, the output depends on the machine. It's not necessary any interaction with the user. The statements presented go in line with the principle of supervised and unsupervised learning detailed in subsection 3.4.4.

The following diagram (Figure 5) illustrates and helps to describe the working of supervised and unsupervised classification techniques. To sum up, in supervised classification and as previously stated, the

result is the assembly of a signature file. Following that, various classification techniques are applied to the created file to classify the image. Unsupervised classification deals with clustering, so no samples are collected for further processing. All work is performed with the help of various algorithms by the computer.

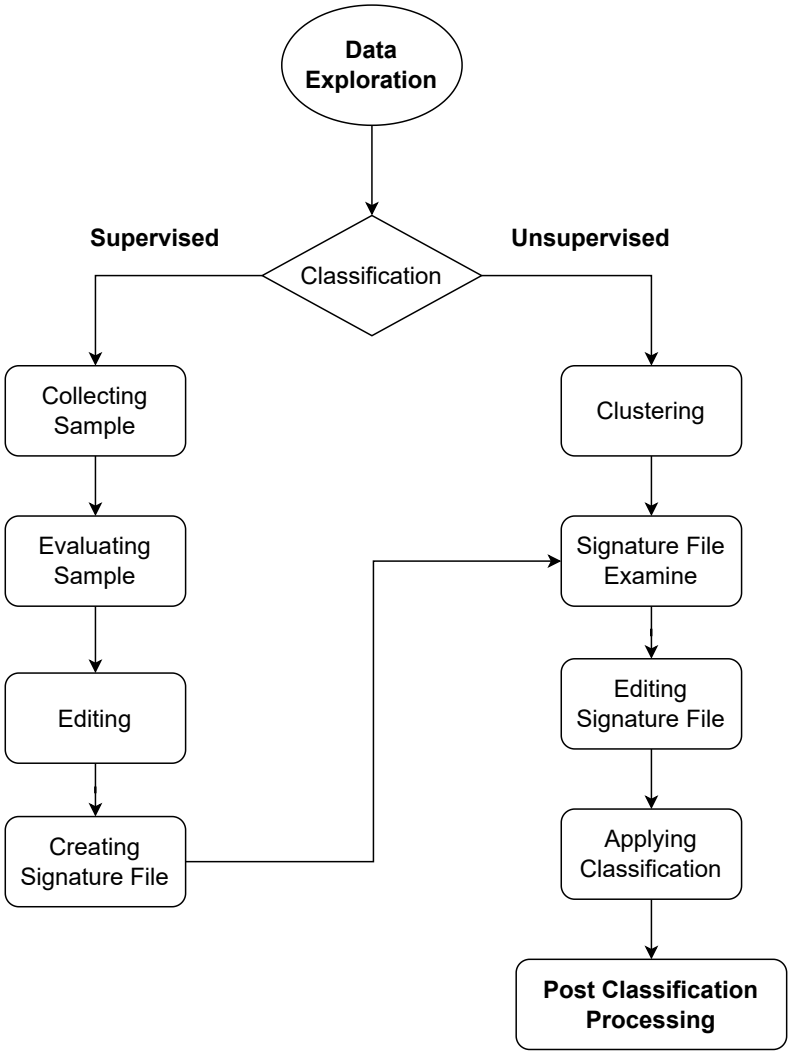


Figure 5: Classification Workflow. Adapted From [23].

3.1.3 Image Restoration

As the name suggests, image restoration is the technique through which a corrupted and noisy image is processed in such a manner to construct the ideal image [24]. There are two types of procedures used to reconstruct an image. One technique is to model the image whose quality is degraded, and the other technique, known as image enhancement, increases the quality of the image by applying various filters [25]. In the subsection 3.1.4 more detailed information about image enhancement is given. Notice that to restore an image correctly is important to have prior knowledge of what may be the cause of degradation.

The following diagram (Figure 6) shows the degradation and restoration activity. To sum up, the restoration of images is achieved via two types of models, namely the degradation model and the restoration model. On the diagram, the original image is represented by the $f(x, y)$. After the degradation has taken place, various functions are applied to restore the image.

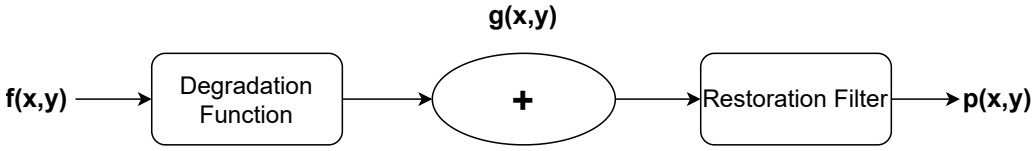


Figure 6: Restoration-Degradation Model.

3.1.4 Image Enhancement

The image enhancement technique helps to improve the quality of the image. This method modifies certain components in images to increase image clarity. Image enhancement is commonly used to analyse an image for feature extraction. Several algorithms are used in this process. As can be seen in Figure 7, there are two different approaches to image enhancement. The spatial domain technique works with pixels. The pixel values are altered to achieve the desired enhancement. It also contains other techniques constantly working dependent on the pixels. The frequency domain technique is used for images that are based on frequency mechanisms and works on the orthogonal conversion of the image rather than the image itself [23].

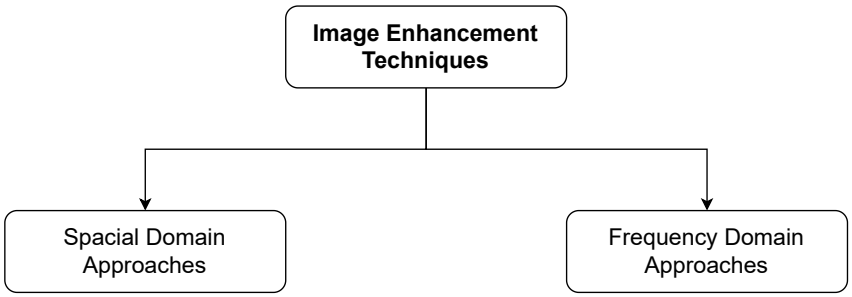


Figure 7: Image Enhancement Techniques. Adapted From [23].

3.2 Image Analysis Techniques

Image analysis is the field of digital image manipulation that differs from image processing. The purpose of the image analysis technique is to emulate human vision, including learning and the ability to make a decision based on input [6]. This method is associated with the DIP processes. However, instead of collecting information qualitatively, these systems extract quantitative information from datasets assembled by a set of images, as will be the case in this dissertation.

Commonly image analysis techniques are applied to images resulting from image processing techniques. The most typical operations associated with image analysis are morphological analysis, measurements, recognition, representation and description. We can also include segmentation in this group of operations. Regarding this study, the recognition based on object segmentation will be used for assigning a label to objects (e.g. microglial cells). Another relevant aspect to point out, and also related to the main goal of this dissertation, is that morphological analysis relies commonly on the geometric aspects of an object. This means that the parameters related to object morphology such as diameter, area, number, perimeter length, roundness, and extension [14], fit perfectly in the area of image analysis of microglial cells. Besides the most basic mathematical morphological operations, three types of operations characterize the morphological analysis technique. Regarding this study we are going to focus on morphological analysis applicable to binary images, usually derived from thresholding. Operations for size and shape are the most common and easiest morphological operations to use and are related to the properties, local shapes and sizes of objects in images. There are several morphological operators to aid in image analysis. Knowing that we want to quantify the number of microglial cells from brain images the ones that are important to point out are "Dilation", "Erosion", "Opening and Closing", "Hit-or-miss Transform", and "Boundary Extraction". Next, follows a brief description of each of the mentioned operators.

Dilation Dilation is the operation in which object edges are expanded. The central pixel of the structural element (which in our study will be the nucleus of the microglial cell) cycles through all pixels of the target object. This results in a wider object (wider nucleus). Related to this increase in size is the number of pixels of the structural element, as they exceed the limits of the target object when the centre pixel reaches the edge of the target object [26].

Erosion Erosion is the exact opposite of the dilation procedure. Erosion causes a contraction of the edges of an object. The object is reduced according to the shape and size of the structural element. Once the limit of this element reaches the boundary of the object, the pixels between the object boundary and the central pixel of the structural element drop out from the constitution of the object. It is important to point out that once the centre of an element passes through all the pixels of the target object this results in the contraction of limits [26].

Opening & Closing Opening and closing operations are related and are intrinsic to dilation and erosion techniques. The output of opening an object is the same as the result of erosion followed by dilation of an object, causing smoothed contours, broken narrow isthmuses and eliminated small islands and sharp peaks. Analogously, the output obtained by closing an object is the same result that we can obtain with dilation followed by erosion of an object [27].

Hit-or-miss Transform Hit-or-miss transform is an iterative morphological shape detection tool. This operation involves several basic operations such as erosion, complement, intersection and difference. After

several iterations where each of these techniques is implemented, the final result includes the coordinates of the object of interest in the image [27].

Boundary Extraction Boundary extraction is the operation that returns a region of pixels corresponding to the boundary of an object of interest. The quality of the resulting boundary is related to the shape of the structural element since the result of this operation is equal to the difference between the original object and the same object after erosion [27].

3.3 Classic Methods for Automatic Cell Counting

Conventional cell counting involves a specific set of tools and devices developed for that purpose. This process is tedious, time-consuming, and inaccurate due to operator-dependent biases. Most of the cell counting processes to this date are manual. However, because cell counting is an important procedure routine that in some cases may help in the detection of illness, various study reports are focusing on the experience of the development of image processing programs to automate cell counting. This makes the cell quantification process more time-efficient and reduces error [4, 28]. It is important to point out that some of these programs, developed with the aim of automatic cell counting, require specific settings on an image to obtain a reasonable accuracy [4]. This is where the image processing and analysis techniques come in.

If an image of the cells spread in one layer on a known area is available, software solutions for automatic cell counting are applicable. Some software have proven to be suitable for automatic cell quantification, such as Cell Profiler, CellC, Cell-Counter, and OpenCFU. Another important piece of software for this matter, and more related to the selected approach based on a more classical methodology, for automatic quantification of cells are ImageJ and FIJI. Both are open-source image processing software, and FIJI is a package based on ImageJ. It is important to point out that these manual standalone cell counting assistants, plug-ins, and guides facilitate cell counting by replacing the manual clicker with multiple digital counters or placing a semi-transparent grid over the image to help with focus. ImageJ is the platform of choice for image processing and automatic cell counting because it has several useful tools. For this study, the choice of ImageJ for the implementation of an automatic cell quantification solution, using a more classic approach, was based a lot on the premise stated before and on the fact that allows studying and designing a solution where concrete and acceptable results are expected. We can find more detailed information about what is ImageJ in subsection 3.3.1.

One option to solve the problem raised can be the "Analyze Particles" method combined with image thresholding. Thresholding an image allows separating the background from the objects of interest, which can then be counted by the "Analyze Particles" function. An alternative can be the "Find Maxima" method of ImageJ. This functionality determines and counts the local intensity maxima in an image. To sum up, with this method each pixel with the average of its 3 x 3 neighbourhood, eliminates small imperfections

of high intensity that contribute to false positives. This can be extremely important for the identification of microglial cells from brain images for the previous quantification [29]. A small review of related work and possible approaches to the raised problem is presented in subsection 3.3.3. The work developed by the scientific community will allow the development and understanding of the best approach to quantify objects with ImageJ.

3.3.1 ImageJ

Until very recently, image processing and analysis were only possible through very expensive commercial tools or even through software packages developed by those who needed them. Furthermore, the acquisition and storage of digital pictures to an image processing system have never been as commonplace as it is today. The ubiquity of digital technology has made digital images part of numerous areas like medicine. In fact, with the increasing flow of the acquisition of digital images, new software packages have been developed, allowing users to manipulate and process digital images. Taking this into account, finer distinctions of certain concepts related to digital image processing performance appeared [30]. As stated in section 3.1, image processing is the conception, development, and improvement of digital images. As previously mentioned, in section 3.2, image analysis comprises all the techniques with the purpose of extracting meaningful information about picture contents.

After having previously introduced image processing and analysis techniques, let's talk about and present ImageJ. ImageJ is a software package containing many functionalities that allow the application of image processing and analysis techniques to images. This software can compute areas and statistics of pixel values, measure distances or even create histograms and line profile plots. With ImageJ, we can do the most basic image processing techniques such as background subtraction, brightness and contrast adjustment, image type conversion, smoothing, sharpening, filtering, and binarization. It also contains other basic features for basic manipulations like geometric transformations such as scaling, zooming and rotation. It is important to point out that it can display, edit, analyse, process, save and print 8–bit, 16–bit and 32–bit images, as well as reads files in the TIFF, GIF, JPEG, BMP, DICOM, FITS format, and supports stacks, that is, a series of images in a single window [31]. ImageJ is a java-based program developed at the U.S. National Institutes of Health. Available on the Internet for the public domain, meaning that its source code is openly available and its use is license-free (at `https://imagej.nih.gov/ij/download.html`) for Mac OS X, Linux and Windows. Another important detail is that ImageJ was designed with an open architecture that provides extensibility via java plugins [32]. If needed, we can use open-source code to install additional plugins related to the automatic quantification of cells.

As intended, when we do a literature search and review to fundament a state-of-the-art and start grounding some foundations for future work, ImageJ was installed, and the first contacts with the software were made. The following subsection was developed to make a simple presentation about the ImageJ software. Therefore, a brief presentation about how image processing with ImageJ is given, by presenting some menu commands as well as a brief description of its features. More information about ImageJ

documentation is available at https://imagej.nih.gov/ij/docs/guide/.

3.3.2 Image Processing with ImageJ

According to the information presented above, related to the characteristics of ImageJ, we can say that this software developed a large ecosystem all over the years. ImageJ has several plugins, and they are highly varied and are capable of modifying an existing function or introducing a brand new one due to the fact of being an open architecture software that provides extensibility as it is an open-source tool [33]. Taking into account the vast list of projects that function within ImageJ it is important to focus on the features that distinguish it from others and why it is the software chosen to be the basis for the study of a solution to the problem presented. The most obvious distinction is its simplicity in image processing. With ImageJ, applying image processing tasks is an extremely simple process because it is a very simple and easy-to-understand graphical interface, which makes the processing task much easier.

Like most of the existing programs, ImageJ provides users with a graphical interface containing a menu bar. The program main window is only a bar that contains the menu commands, a toolbar, a status bar and a progress bar as can be seen in Figure 8.

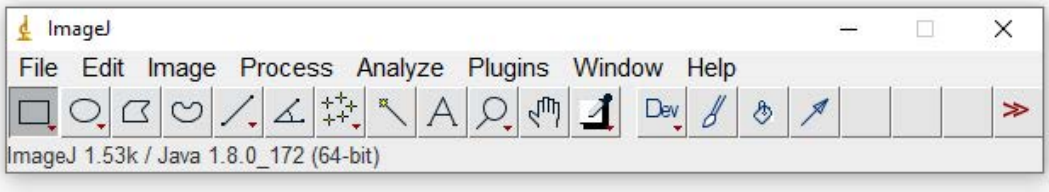


Figure 8: ImageJ Main Window

Additional windows can be opened to show images, histograms, plots and results. The toolbar contains the basic tools to make the following selections:

1. Rectangular and Rounded Rectangular;
2. Oval, Elliptical and Brush;
3. Polygon;
4. Freehand;
5. Straight, Segmented, Freehand Arrow Line;
6. Angle;
7. Point and Multi-point.

Besides this, ImageJ has seven free slots to select any tool from the 60-plus tools offered by the software. There are also 15-plus toolsets available online that can be installed later if necessary. The menu bar has eight menus. The file menu contains the most basic operations like opening and saving. Then, the edit menu allows editing and drawing, as well as contains operations for all global configurations. The following menu, the image menu, has on it the operations for image modification and conversion including geometric transformations. The process menu includes the options for image processing, and the analyze menu has all the operations related to image analysis, such as statistical measurements, histograms and profile plotting. The plugins menu contains all the commands for creating, editing and managing add-ons, as well as the lists of all installed plugins and scripts. Finally, the window and help menu are pretty much self-explanatory as they feature the selection of windows, the documentation and version information, respectively. Regarding the main purpose of this dissertation and after this short exploration and explanation of ImageJ menus and features, the ones we are going to focus more on are the image, process and analyze menus because they are the ones related to image processing and analysis. These menus will make it possible to implement an approach for automatic cell quantification.

Image Menu The image menu is composed of several sub-menus, like type, adjust, colour, stacks and hyper-stacks, crop, duplicate, rename, scale, transform, zoom, overlay and lookup tables. The sub-menu type determines the type of the active image. As stated in subsection 3.3.1, the image types supported are 8-bit, 16-bit, 32-bit, 8-bit colour, RGB colour, RGB stack and HSB stack. The adjust sub-menu contains commands that adjust brightness/contrast, adjust threshold levels, window levelling, colour balance, image size and canvas size. This sub-menu will be extremely important for the process of image processing once as it will assist in the preparation and adjustment of brain images. As expected, the colour sub-menu contains commands that allow the user to manipulate colour. The ImageJ can handle three types of colour images: pseudocolour images, RGB images and composite images. The stacks sub-menu allows various manipulations of image stacks. Image stacks are sets of images displayed in a single window. These images should be spatially or temporally related, and all the images must have the same size and bit depth. The constituent images of a stack are called slices. It is important to point out that most of the existing commands in ImageJ have the option of processing all the slices in a stack. Crop and duplicate sub-menus allow cutting or duplicating an image or stack. Rename allows the user to change the title of an active image or stack. The scale sub-menu opens a dialogue box allowing the user to resize or scale an image or a selected area. The transform sub-menu enables the user to apply geometric transformations to an image. It is possible to control how the current image is displayed through the zoom sub-menu. To conclude, the lookup tables submenu contains a selection of colour lookup tables that can be applied to grayscale images to produce pseudocolour images.

Process Menu The process menu is composed and has the following options that allow implementing image processing techniques: smooth and sharpen, find edges and maxima, enhance contrast, noise, shadows, binary, filters, bach, image calculator and subtract background. Smooth is a filtering function

which blurs an image or a selected area. Sharpen is a command to a processing technique that accentuates and enhances the detail of an image, particularly the edges of objects. The filter finds edges and uses a detector to highlight sharp changes in intensity in the active image or selection. The noise sub-menu contains commands to add or remove noise from images. The shadows sub-menu have a group of operations to produce a shadow effect in image objects. The binary sub-menu has commands for morphological analyses of binary images. As expected, the filters sub-menu contains several filters differing from each other essentially by the method applied. Related to methods applied to filtering we can find the convolution, gaussian, median, mean, and unsharp methods. The image calculator performs arithmetic operations like addition, subtraction, multiplication, divide, between two images.

Subtracting background is useful for images with high backgrounds, but even images with a lower background may benefit from the subtraction of background. The process is based on the "rolling ball"concept in which a ball with a given radius rolls over the bottom surface of a three-dimensional grayscale 2D image. As stated, this subsection was developed to have the first contact with ImageJ which will be used for the automatic quantification of microglial cells when applied a more classical approach. For that, and as an example similar to the case study, available at `http://imagej.nih.gov/ij/images`, an image of a cell colony was selected, to evidence the benefits of image processing. In Figure 9, to the left, we can visualize the original image. To the right, we can visualize the image whiteout the background after applying the "rolling ball"effect with 50 pixels of radius over the image surface. The ability to choose to make the subtraction of dark backgrounds with lighter objects and choose to create a background (not subtract) are features present in ImageJ that make image processing a simpler task, and will help with the identification of microglial cells for previous automatic quantification.

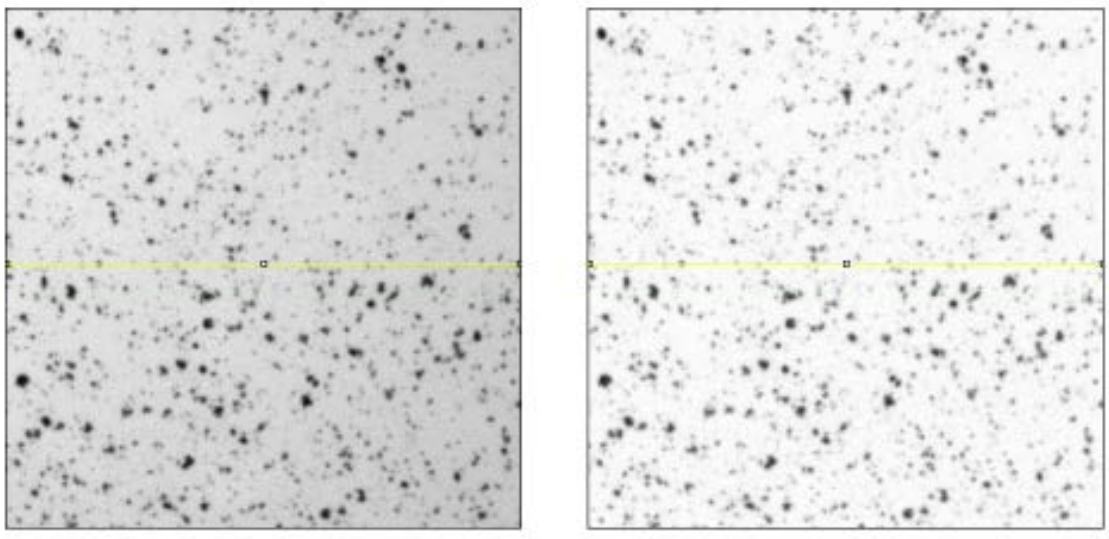


Figure 9: Background Subtraction on a Cell Colony Image

Analyze Menu The analyze menu is it is the menu that allows carrying out the last settings before the actual counting process. As stated, a very peculiar feature of ImageJ is the possibility that it has for counting objects, such as the number of cells from 2D images. Automatic counting is a program application that is very interesting for its speed, ease of execution, and low error. This is why ImageJ was selected and is the perfect solution to solve the problem of automatic quantification of microglial cells. Before the automatic counting of objects in an image (e.g., the cell colony image presented in Figure 9), it should be applied to the image set of processes to make the automatic quantification more precise. The one applied for demonstration purposes, background subtraction is one of those processes. As expected, the results are higher evidence of the cells, as was an increased gap between the background pixels and cell pixels. Before automatic counting, thresholding is a fundamental procedure. This process is image binarization. A manual adjustment with the sliders may be required to decrease the overlap, but is also possible to rely on the automatic adjustment provided by ImageJ. The result is an image in which the white areas have no interest in quantification.

After applying and selecting the threshold, automatic counting is done with the analyze particles method contained in the analyze menu. As evidenced in Figure 10, the menu window for particle analysis is opened. It is important to notice that a prior image scaling was required. On the first line is specified the range of the size of objects to be counted. Following that, was set which objects are included in the counting based on their circularity. This is possible by varying the range of circularity, where the minimum extent (0) corresponds to a straight line and the maximum measure (1) corresponds to a perfect circle. A particularity of ImageJ is the possibility of applying operations to selected areas or regions of an image. It is important to point out that the analyze particles operation is truly useful in counting objects of different colours in RGB images. This feature allows the differentiation of different types of objects and provides count results for different types.

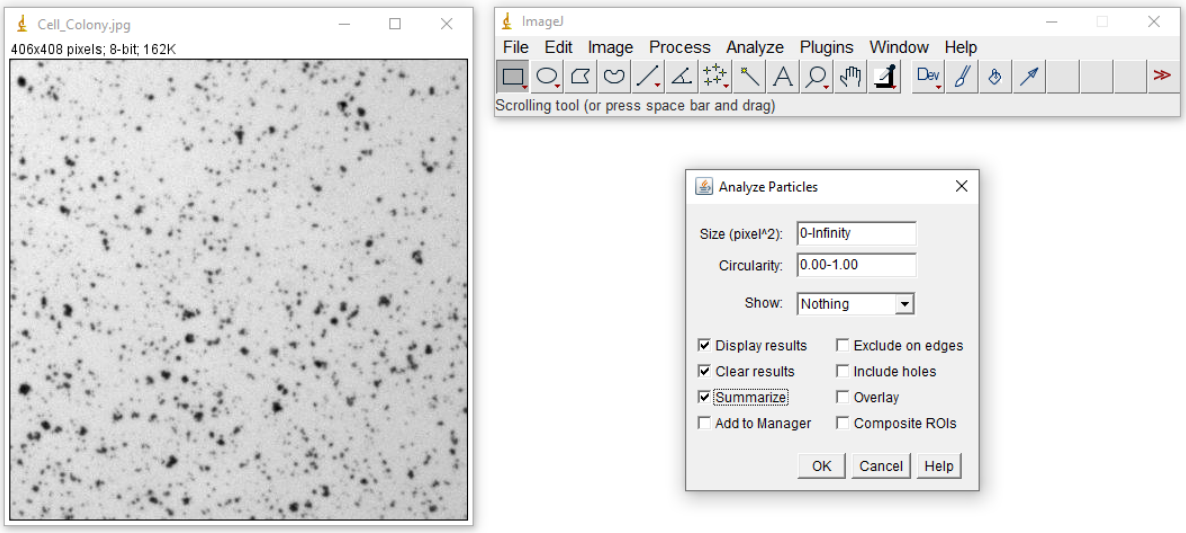


Figure 10: Analyze Particles for Automatic Cell Quantification

3.3.3 Object Counting with ImageJ

The big goal of this dissertation is to perform automatic counts of cells, namely microglial cells. The quantification of cells can be done in two different ways, manually and automatically. Regarding the last one, two methodologies are associated with it, a more classical and a deep learning-based one. The objective of this subsection is to review, within relevant work and literature, different classical approaches to the automatic quantification of cells problem. In total, three articles will be brought for discussion to present some aspects regarding how to do object counting with ImageJ. Some of those will be taken into consideration when designing a final solution.

Quantifying Microglia Morphology from Photomicrographs of Immunohistochemistry Prepared Tissue Using ImageJ

The article produced by Young et al. [34] consists of a set of steps and ImageJ protocols used by the research team to convert fluorescence and bright-field photomicrographs into binary and skeletonized images. Microglial cells are captured in a fluorescent format, and to make certain operations with those images, i.e., count, they need to be converted to a binary format. In addition to the objective of counting these cells, in this study, there was also the objective of analysing their morphology.

Introduction In the referred study, the authors detailed in a stepwise way how they use ImageJ plugins to summarize microglia morphology. The analysis techniques were implemented with AnalyzeSkeleton (2D/Three-Dimensional (3D)) and FracLac16 plugins. The first quantifies microglial cell structures, and the other quantifies microglial shapes. These plugins offer a rapid analysis of microglia ramifications within entire photomicrographs. They stated that the use of both tools is not redundant, as cell ramification is complementary to cell complexity. In addition to the statement information, a protocol for counting these cells was also developed.

Experimental Work In terms of the developed work, in addition to the objective of counting microglial cells, in this study, there was also the objective of analysing their morphology. Therefore, of all the steps performed, we will only focus on two, step 3 (Imaging) and step 4 (Skeleton Analysis) since they are the ones that make it possible to elaborate an approach to the quantification of microglial cells and are related to the theme of this dissertation. The samples used of microglial cells contain the IBA1 marker. Therefore, in step 3 the goal was to separate channels since cells are only visible in the red channel. It was possible to perform this action using ImageJ's channel separation feature. To perform the automated count of these cells, in step 4 they used the unsharp mask filter to increase the contrast on the image. Next, they needed to adjust the threshold of the image to convert it into a binary format. Following this protocol and using the analyze particles functionality, they were able to automatically count cells using ImageJ.

Results As stated, several steps were implemented to represent microglia morphology according to metrics such as cell ramification, complexity, and shape. The developed protocol steps that helped in the

identification of cells as the image noise was removed, and only the cells themselves were left for analysis and counting. Throughout the results obtained, it was possible to prove that ImageJ protocols make microglia morphology quantification available to all laboratories as the platform and related plugins are open-source. Although their main objective was to enhance binary, skeleton, and outline representations of complete photomicrographs and single cells, they were able to make easier the automatic quantification of cells process. It is important to point out that supplementary modifications can be easily made to the protocol depending on image quality and on the actions to reduce noise. Thus, through the results obtained, we conclude that some of the steps will need to be taken into consideration to design a solution for the problem presented in this dissertation.

Discussion and Conclusions To sum up, this paper provided a general overview of a developed protocol with recommended ImageJ plugins for automatically accessing the morphology and quantifying microglial cells. The main goal of the protocol is to convert fluorescence photomicrographs into binary skeletonized images. Additionally, they concluded that with the use of this protocol, microglia quantification is accessible to all laboratories, as the plugins used are open-source. This article was also able to verify that it is possible to carry out cell counting in a more automated way, thus avoiding manual counting approaches and processes.

Assessment of Cell Counting Method based on Image Processing for a Microalga Culture

In Dökümcüoğlu et al. [35] the main objective aimed to illustrate the effectiveness of an image processing approach for the automatic counting of cells in a microalga culture. Therefore, they also develop a protocol with ImageJ to automatically quantify cells.

Introduction In this study, the authors also detailed in a stepwise way how they automatically quantify cells in a microalga culture. Their main goal was to prove the usefulness of an image-processing approach for counting cells. Thus, to attest usefulness of image-processing software for cell counts, they compared the performance obtained between the ImageJ cell counts, Utermöhl cell counts, which is a more manual procedure for the quantification of cells, and Optical Density measurements.

Experimental Work In terms of the outcome, we are only going to focus on the ImageJ one. Similar to the previous solution, they developed within ImageJ a protocol to automatically quantify cells. They started by calibrating their image and then converting it to 8 bits to minimize colour variation. For better visualization of dark-coloured cells, they subtracted the background through the background subtraction functionality within ImageJ. To obtain a better and clear image, they also applied the rolling ball radius to reduce the noise in their images. To convert the image into a binary format they also adjust the threshold, and to obtain more accurate results, they fill the holes in cell nuclei. Finally, they used the analyze particles functionality to be able to automatically count cells.

Results The results obtained attest to the benefit of an image-processing approach to quantify cells. The plots presented in the article represent the results of regression analysis between optical density and cell counting methods. The first one is the analysis between the Thoma chamber and optical density, the second is the Utermöhl chamber and optical density, and the last one is ImageJ counts and optical density. Through the analysis of results, they were able to conclude ImageJ cell counts were finished in a quarter of the time used for manual cell counting under the microscope. In this way, they attested ImageJ can improve and automate the results of the existing automatic counting methods and increase the associated work speed and reliability.

Discussion and Conclusions The authors were able to demonstrate the usefulness of image-processing approaches for the quantification of cells. In their case, the samples were a microalga culture. Their approach can be easily applied to the quantification of microglial cells problem. They are similar because the brain image contains multiple glial cells, like in cell culture. To conclude, they also evidence in this study that resorting to ImageJ allows the operator to complete the counting process four times faster than a manual count, with similar accuracy.

Automated Segmentation and Analysis of Retinal Microglia within ImageJ

In Ash et al. [36] a segmentation routine was proposed to perform automated segmentation and cell counting. Seeking to automate microglia counts, they concluded that few algorithms exist for retina microglia count. The experimental work within the FIJI-ImageJ ecosystem originated a new segmentation routine to perform automated segmentation and cell counting in retinal microglia. They showed that they can perform cell counts with similar accuracy to manual systems.

Introduction Through this study, and knowing that microglia are immune cells of the CNS capable of migrating in response to injury, the author's identified that algorithms aiming to automate microglia counts and morphological analysis are becoming increasingly popular. Few exist that are acceptable for use within the retina and manual analysis remains dominant. With FIJI-ImageJ they performed automated segmentation and cell counting and evidence that their procedure routine can accomplish counts with accuracy comparable to manual observers using the I307N Rho model.

Experimental Work In terms of the experimental work, they started with raw images as input. Then they preprocessed those images by normalizing stack intensities followed by the application of a rolling ball, median filter, and Gaussian smoothing routines. The next step was to convert the image to a binary format. The fourth step was cell somas which were identified by morphology using an existing algorithm, and their overlay with the candidate cell masks. The fifth step was labelling. After the application of the watershed algorithm, they were able to identify distinct cells as this algorithm separates with one pixel what he considers to be one or more cells together. Finally, they indicated high overall fidelity with occasional undetected cells and dendrites. The quantification was performed after all these steps were conducted.

Results The results obtained after the application of the image analysis protocol are images of cells where the noise is reduced and the cells are evident, which undoubtedly facilitates the counting process. The developed procedure segmented contender cell masks by watershed regarding the overlaid cell markers. This method allowed a more accurate definition of microglial morphology. Finally, cell counts were obtained using the labelled image produced within FIJI-ImageJ. Once again, through the results obtained in the analysis and counting process, the team attested to the benefits of automated cell counting processes when compared to more manual processes.

Discussion and Conclusions The authors implemented, within the FIJI-ImageJ ecosystem, a new segmentation routine to perform automated segmentation and cell counting in retinal microglia. As the algorithms that automate microglia counts are increasing in popularity, they conclude that few of those are adequate for segmentation and cell counting of retina microglia. Therefore, their approach built entirely with open-source software, addresses the presented problem. Throughout the results, they showed that their routine can perform cell counts with similar accuracy to manual counting but faster, thus evidencing the benefit of automatic approaches to the problem of cell counts.

3.4 Deep Learning for Automatic Cell Counting

According to the information stated above the identification and counting of the number of cells from an image are one of the biggest tasks for biomedical image analysis and medical diagnoses [37]. Cell counting, in particular microglial cell counting, is conducted in this study because these cells are responsible for fundamental physiological and pathological processes. Therefore, its identification and quantification may help detect a serious illness. The major weakness of the state-of-the-art techniques for cell counting is the counting dependence on low throughput technology that requires manual counting done by specialists, and adjacent to this, there are high labour costs, error-prone data collection, user subjectivity and fatigue. In recent years, deep learning-based approaches evidenced promising performance in various image analysis tasks, such as classification, detection and segmentation. The most important thing to notice is that this approach has shown similar accuracy to manual counting but a significant enhancement in reproducibility, throughput efficiency and reduced error from human factors.

Regarding the cell counting problem, with this approach, the problem can be categorized as detection-based counting and regression-based counting [38]. The first approach requires the detection or segmentation of every cell before cell counting, which implies a supervised learning process. To convert a counting task into a segmentation task, cell annotation is needed, to train the detection or segmentation model. Each cell is detected one by one through the object detection model, and then the counter takes the detected cells and produces the counting results. To sum up, the first step of this approach is the identification of the cell-like candidate region, the second step is the evaluation of the candidate, which can be done by an Support Vector Machine (SVM). The last step is the selection of a non-overlapping region.

Currently, more studies have been focused on regression-based cell counting as they avoid the challenging task of detection or segmentation of single cells because they generate cell density or cell count directly from the images [38]. The Convolutional Neural Networks (CNNs) models are been applied and modified using the Euclidean loss function by taking the total number of cells as the annotation information [39]. The number of cells within a certain region is obtained via the integration of the density map.

3.4.1 Machine Learning

To better understand what Machine Learning (ML) is, there are certain basic concepts of Artificial Intelligence (AI) that we must first comprehend. The term AI is commonly used to refer to all computer programs that can think like humans, in other words, AI is defined as a computer program that exhibits human-like cognitive ability. Major AI researchers and books define this field as "the study and design of intelligent agents", being the intelligent agents the systems that understand the environment and take certain actions to obtain success [40]. Any computer program that shows the stated characteristics, such as self-improvement, learning from inferring, or even basic human tasks, such as image recognition [41, 42], is considered to be a form of AI. The field of AI includes within it the sub-fields of ML and Deep Learning (DL), each with its specific characteristics, but in the end, they are all related. It is important to point out that ML approaches are more probabilistic, which means that the output can be explained, thereby ruling out the black-box nature of AI, unlike DL approaches that are more deterministic.

ML is an application of AI [43], in and of itself, is a relatively simple concept. The idea behind ML is that machines should be given access to data and learn specific tasks by themselves to make accurate predictions or be able to behave intelligently without being explicitly programmed. They learn to do better in the future based on past experiments. For that, ML exploits computing systems that learn and predict from data, which can be examples, direct experience, or instruction [44]. Because of this, the main advantage of ML over human learning is the ability to consume huge amounts of data, learn from it, and detect and analyse patterns that outshine human capabilities. The data consists of a set of samples that usually represent the observed variables. It is important to point out that in some cases like the one presented in this dissertation, the set of observed variables is composed of images. As explained, ML algorithms can improve through training from that data, to improve their predictions. Most of those algorithms have certain settings, normally denominated as hyperparameters. During training, those hyperparameters help to control the algorithm's behaviour. It is noteworthy that a subset of the training set is the one used to choose the hyperparameters for the model. The selection process can be made by trial and error because the validation set isn't used to train the ML algorithm [45].

To implement an intelligent system learning is required. As stated, a system must be able to perform actions that are associated with intelligence. Despite the complexity of the implementation of learning, this is a subject of great interest to the scientific community due to their large number of application domains, such as medical diagnosis. Today, three prominent methods are being used to train these algorithms.

These training methods, most commonly known as learning paradigms are supervised, unsupervised and reinforcement learning. In subsection 3.4.4, we can find more detailed information about the learning paradigms.

3.4.2 Deep Learning

We can say that DL is a more specialized version of ML because utilizes more complex methods for difficult problems [46]. The term "deep" helps to justify the previous statement as it refers to the number of layers in the network. DL stands as a sub-discipline within ML, that according to what has already been explained is a subset of AI. Despite being a sub-discipline of ML, DL algorithms differ from ML algorithms in their ability to learn from unstructured and unlabelled data. ML algorithms require labelled data, which is a task performed to make data readable for the program. DL algorithms can process raw data without the need for labelling [47]. DL specifically is the use of the concept known as Neural Networks (NN), whereby computers emulate the systems of neurons, similar to those found in the brain, to learn and work [44]. NN and DL for many specialists represent the way forward for AI as we know it, as they will pave the way for human-like AI shortly. NN and other subjects related to the fundamentals of DL are described in more detail in subsection 3.4.3. In DL, the complexity is described in the relationship that variables share. A system that relies on or uses simple concepts and variable relationships to learn more complex concepts is known as a DL algorithm. This goes to find and substantiate the claim DL algorithms differ from ML algorithms in their ability to learn from unstructured and unlabelled data [48].

To sum up this subsection and highlight some of the most important ideas, DL stands out mainly because of these three factors: robustness, generalization capacity, and scalability. A model can automatically learn from raw data, or in other words, can understand the most important features even in the presence of noise [49]. The model's performance tends to improve when trained more and more.

3.4.3 Fundamentals of Deep Learning

Related to the study developed with this dissertation, some theoretical fundamentals of DL are essential to point out, to more easily understand the reasons behind the choice of this approach. A notion of artificial neurons, moving on to how they can be assembled in a network, and finally introducing CNNs are the theoretical foundations of DL presented below.

Artificial Neuron

The human brain is considered a massive network and has approximately 86 billion neurons [50]. The neuron represents the basic computational unit of the brain. Usually, neuron inputs come from dendrites which are connected to other neurons. The input signals are all processed together according to the strength or weight of their connection. The newly generated signal is then sent through the output axon to other neurons' dendrites. In Figure 11, we can see a representation of a biological neuron. This one is in genesis and inspired the creation of the mathematical model for artificial neurons.

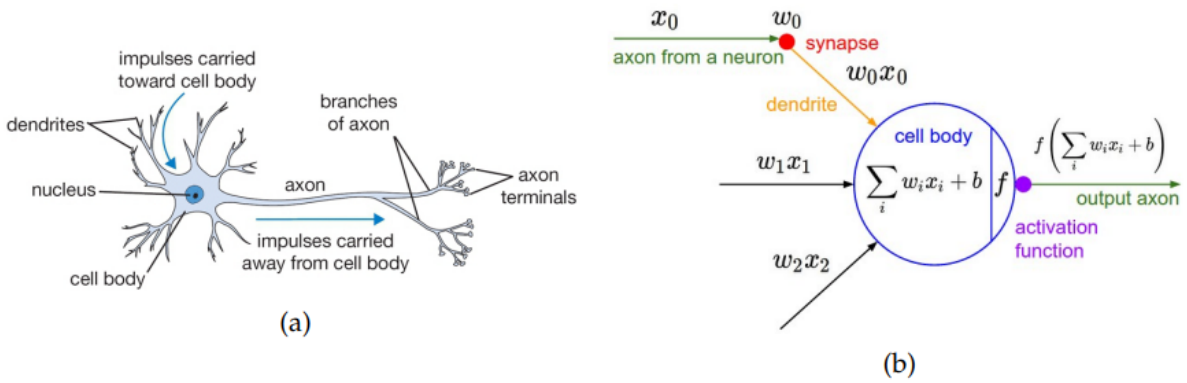


Figure 11: Representation of a Biological and an Artificial Neuron. Adapted from [50].

The artificial neurons are the base element of ANNs. They transform a set of inputs into a single value using a weighted sum since each input has a respective weight. The neuron output values are calculated by applying an activation function. In Figure 12, we can see the representation of the most commonly used activation functions. Following, a small definition and description of these activation functions are presented.

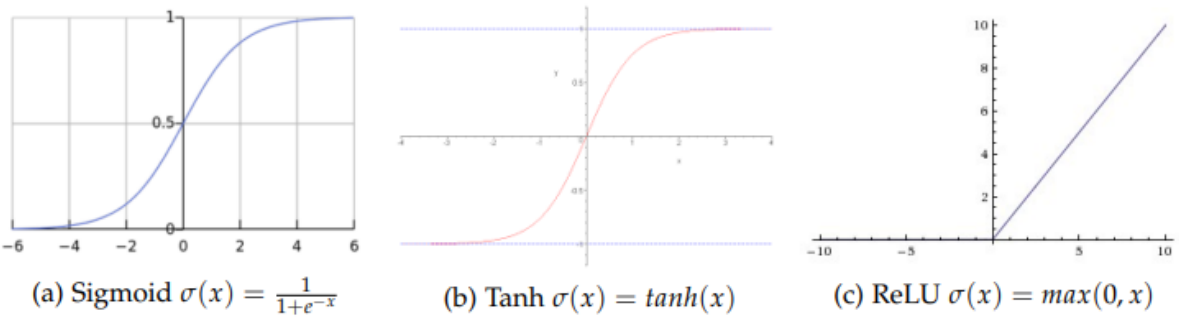


Figure 12: Representation of Activation Functions. Source [50].

Sigmoid The function outputs are values between 0 and 1. It saturates low input values closer to 0 and high input values near 1. However, this activation function has a small drawback of not being zero-centred, once the output is always positive. That said, the gradient on the weights will become all positive or negative, according to the signal [50].

Hyperbolic tangent This activation function, also known as Tanh, returns values between -1 and 1. Similar to the sigmoid function, this one also saturates high and low input values. However, there are differences between the two of them. The main difference is that for low input values the output will tend to be -1, and, for high inputs, it will tend to be 1 [50].

ReLU This function operates with a threshold of zero. The negative input values are set to zero and the remaining values are left unaltered. This activation function is based on computational simplicity, and for that accelerates training. However, the ReLU definition is fragile and many neurons can irreversibly die during training, in special, with high learning rates [50].

Artificial Neural Networks

The term ANNs is usually used to characterize the mathematical model composed of a collection of artificial neurons. As stated in the previous section, these neurons are interlinked in a network by layers to learn more complex data relations [49]. Feedforward Neural Networks (FNNs) and Recurrent Neural Networks (RNNs) stand as two types of ANNs. The FNNs are characterized as being an acyclic graph where the data only can move forward. Unlike these, with RNNs the data/information can flow in any direction, including to the same layer. To sum up, RNNs are an extended representation of FNNs with the addition of feedback connections [49]. As we can see in Figure 13 (a) contains three input neurons, two hidden layers (both with four neurons) and an output layer with a single neuron. Traditional FNNs models can be disposed into fully-connected layers of neurons and the data moves forward. On these layers, only neurons between two layers are fully pairwise connected. That means that neurons within a single layer don't have connections between themselves [50].

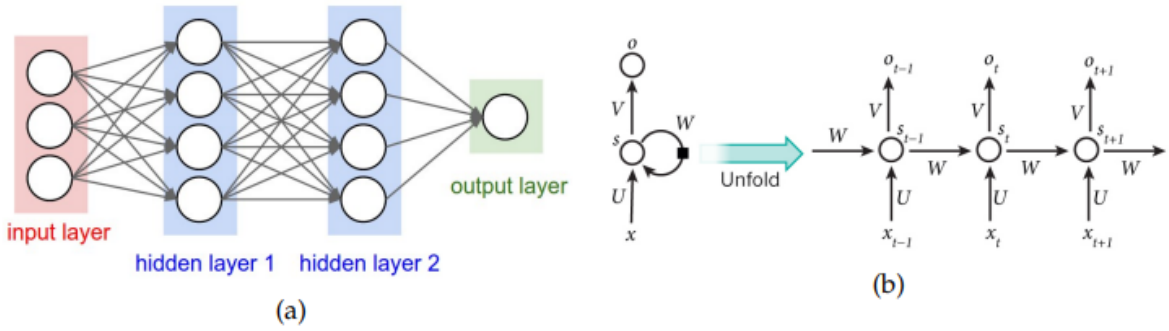


Figure 13: Representation of two types of Artificial Neural Networks. Adapted From [50].

Traditionally FNNs are structured in three sections:

- An input layer that receives the data. The data needs to be in a vectorized representation, for instance, and focusing on the problem presented with this dissertation, to deal with images, they must be converted to a One-Dimensional representation, just like a vector;
- One or several hidden layers, which are composed of artificial neurons to extract non-linear features from the data;
- An output layer that combines a set of non-linear features learned on the previous layers and outputs.

The RNNs have the purpose of processing sequential data [49]. The term recurrent is used to refer to the ability that these ANNs have to execute the same job for every element present in a sequence. Figure 13 (b) shows that for each time-stamp t, the hidden state is calculated by adding the current input with the previous hidden state, in which the function f is the activation function. In the context of supervised learning, which is described in more detail in subsection 3.4.4, on both types of ANNs, the error between the network predictions and ground truth is back-propagated throughout all networks and is normally used to update the network weights to make the predictions more accurate [51].

Convolutional Neural Networks

According to the information presented above, ANNs are composed of a set of artificial neurons that are organized into layers such as the input layer, the hidden layer, and the output layer. CNNs follow the same principle as they derive from ANNs. However, it is important to point out that they differ in one simple detail, as they assume that the input is a set of images. The architecture of CNNs disposes of the neurons differently when compared to ANNs as they look to improve the creation of the model [50]. Neurons are disposed of in three dimensions such as width, height and depth [49]. The idea behind CNNs is to use small ANNs convoluted along with the image to extract relevant information. It is important to point out that this idea differs from a fully connected architecture which demands much more parameters. As a reference, a single fully connected hidden layer requires 120 000 parameters to handle a 200x200 RGB image. This number of parameters is wasteful as leads the model to easily overfit on training [50]. Besides that, with a fully-connect approach, the image pixels are related together. When using smaller CNNs, it is possible to identify local features and add successive convolutions layers. In the Figure, we can better understand how neurons are set on CNNs in comparison to ANNs. Figure 14 displays the traditional representation of ANNs and CNNs.

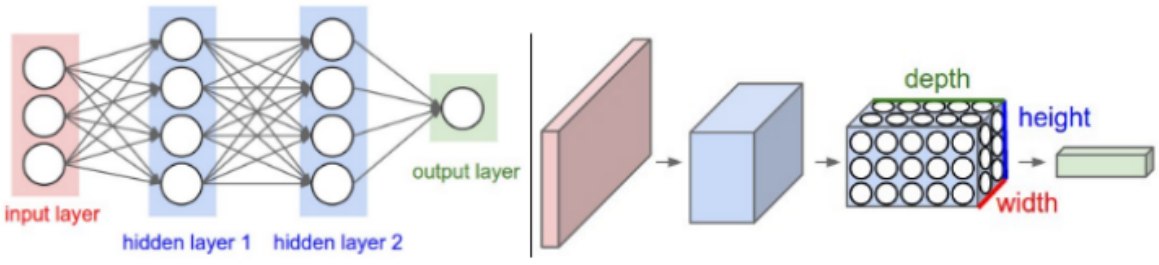


Figure 14: Representation of a Traditional Artificial and Convolutional Neural Networks. Source [50].

The concept of receptive fields is key to ensuring that neurons inside a layer are only connected to a small region of the previous layers and with this avoid the wastefulness of fully-connected neurons. This allows for exploring the local connectivity among neurons. CNNs are capable of generalization on vision problems and help to reduce the number of parameters [50]. As we can see, in Figure 15, typically a

CNNs is structured in two main sections. First, we have the feature extraction followed by the classification process. The main goals of these two sections are:

1. Feature extraction - plays an important role on CNNs as they are responsible for learning patterns. They learn how to extract the most relevant features automatically and as expected when the network grows deeper. This first section of the CNNs is computationally expensive due to convolutional layers. Activation and pooling layers are alternatives that can be used. However, they have the same problem as convolutional layers because they are also computationally expensive [51];

2. Classification - this section of CNNs receive the features extracted in the previous section. It can be composed of one or more fully connected layers as well as some dropout layers between them [52].

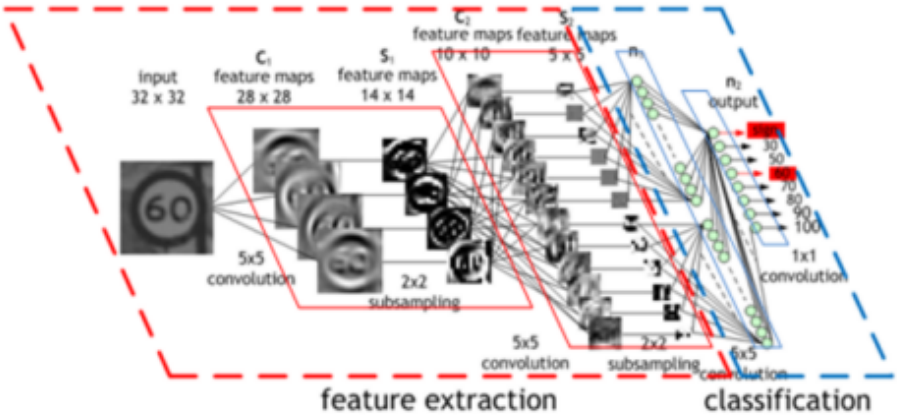


Figure 15: Typical Structure of Convolutional Neural Networks. Adapted From [53].

Convolutional layers Convolutional layers represent the base element of CNNs, as they are responsible for several computational convolutions. The most correct interpretation for convolution is a cross-correlation once the convolutional filter works as a feature detector [49, 51]. Regarding this study, an input image will produce a big output feature map. However, there are other properties and considerations regarding the convolutional layers that need to be pointed out. It doesn't make any sense to fully connect all neurons to all pixels in an image because nearby pixels have a higher probability of being correlated than those that are far apart [50, 53]. This leads to small connections being made between a neuron and a small receptive field that have the same size as the applied filter.

Activation layers Activation layers have the purpose of adding non-linearity properties to the data. As they keep the volume unchanged, they perform an element-wise fixed mathematical operation [50]. These layers are associated with activation functions, the same ones applied to ANNs that are described in this subsection. For activation layers, the simplest activation function that we can use is the ReLU function which makes the training faster without compromising performance.

Pooling layers Pooling layers perform downsampling along the spatial dimensions, in particular width and height. The pooling layers are responsible to shrink areas to a single value, to condense more information. They bring a lot of benefits such as the improvement of computational performance. But it is relevant to point out that a large pooling area may result in data loss [50, 53].

3.4.4 Learning Paradigms

To improve the accuracy of predictive models, ML techniques are required. About a learning paradigm, we can say that it states a particular way or pattern in which something or someone learns. Depending on the nature of the problem being addressed the ML paradigms vary. There are different approaches based on the type and volume of data, which in turn have different amounts and types of supervision in training [44]. Each has its advantages and disadvantages. To better understand their pros and cons, it is very important to realize what kind of data they ingest. ML has two kinds of data, labelled and unlabelled data. Labelled data requires a lot of human labour, to begin with because the input and output must be labelled in a completely machine-readable pattern. In opposition, unlabelled data only have the input or the output in a machine-readable form. In some cases, both can be unlabelled which completely denies the need for human labour.

Supervised Learning A supervised approach usually begins with established data and knowledge about how the data should be classified. The previous statement goes in line with the principle of supervised learning, which requires that the data used in training contains labels that are the outputs for each input. The model has a mapping function, which is formed by an algorithm that differs from problem to problem after being trained to predict the output data for each input [54]. The algorithms are trained using pre-processed examples. Occasionally, patterns are identified in a subset of the data. If we are faced with the case that the model is only fit to represent those patterns, we have an overfitting problem. To prevent this overfitting problem testing needs to be done against labelled data [44]. As stated, supervised learning requires labelled features to define the meaning of the data. For example, there could be millions of images of birds and include an explanation of what each bird is. Then we can create an ML application that distinguishes one bird from another. With labelling, attributes and the meaning of the data are identified and become perceptible to the users. When the labelling of the data is continuous, we are facing a regression problem. When the data comes from a finite set of values, we are facing a classification problem [44]. A typical classification problem is where the model needs to predict the correct categorical output, such as classifying whether a patient has a disease or not or classifying whether an email is spam or not. Given that the data involved in training contains each email labelled, the model must learn how to predict the label, in this case, spam or not spam, in the most accurate way possible for each new email. The other type of supervised learning problem is the regression problem. A machine is trained to predict a specific value, such as the pricing of a house or a market stock [55]. It is important to point out that supervised

training models have broad applicability to a variety of problems. Classification is a typical example of a supervised learning task.

Unsupervised Learning Unsupervised learning differs from supervised learning as it can learn from no results. The model tries to learn without any supervision. A machine is provided with just the input to develop a learning pattern. A computer learns through observation and has to find structures in the data. The model only has available the inputs without any corresponding outputs. This learning paradigm is best suited to problems that require a massive amount of unlabelled data [55]. Algorithms understand the meaning based on data to be able to classify it based on the patterns or clusters that are found. Unsupervised learning algorithms segment the data into groups of examples (clusters) or features. The unlabelled data creates the parameter values. This process allows adding labels to the data. Therefore, we can conclude that unsupervised learning can be the first step before passing the data to a supervised learning process. That can help determine outcomes more quickly than a supervised learning approach [44]. This approach can be divided into three main categories to be best suited to different kinds of problems such as clustering, association rule learning and visualization and dimensionality reduction, which have all common large volumes of unlabelled data. A clustering problem is where we want to discover the inherent groupings in data. To sum up, clustering is grouping up each input such that each one ends in the same cluster or group. An association rule learning problem is where we want to discover rules that describe large portions of our data, for example, people that buy the object X tend to buy object Y. Finally, the visualization and dimensionality reduction problem consist in presenting data so that it is understandable how the data is organized. Dimensionality reduction is discovering which inputs reflect better the general aspects of the data and simplify the dataset without losing information [55].

Reinforcement Learning Reinforcement learning is a behavioural model. To help fundament that statement, we know that reinforcement learning uses a method based on rewards and penalties for each action that it takes to train the model. With that, the user is guided to the best outcome. Reinforcement learning differs from other learning paradigms because it isn't trained with the sample dataset. Therefore, a sequence of successful decisions results in the process being "reinforced" better solve the problem at hand [44]. The most common applications of reinforcement learning are in robotics or game playing. In the case of robotics, a simple example of the application of this type of learning is when we train a robot to navigate a set of stairs. The robot knows that it needs to change its approach to climbing a set of stairs based on the outcome of its actions. The learning algorithm behind all that has to be able to discover an association between the goal of climbing stairs successfully without falling and the sequence of events that lead to the outcome. Well-known examples of the use of this paradigm are the AlphaGo. By implementing reinforcement learning algorithms, the robot or program learned how to do a certain action.

3.4.5 Variational Autoencoders & Generative Adversarial Networks

DL can easily extract high-level information and features that other methods composed of simple representations can't. Whether the data is labelled or not there are two possible approaches. As stated previously, in supervised learning is required that the dataset features must be labelled. When facing unsupervised learning tasks, labelling data is not required. The algorithms associated with unsupervised learning explore the structure of the data. However, in some cases, some of these models have limitations because they do not have good generalizations. To solve this problem, Variational Autoencoders (VAEs) e Generative Adversarial Networks (GANs) can be a better alternative [56].

Variational Autoencoders VAEs provide probabilistic descriptions of observations in latent spaces [57]. VAEs have ANNs architecture. Therefore, they belong to the families of probabilistic graphical models and variational bayesian methods. Unlike more common approaches, such as NN as regressors or classifiers, VAEs are a powerful generative model. Using generative models, we could generate a random model or a new output. The typical setup of VAEs, described in Figure 16, consists of a pair of networks: the encoder and the decoder. The encoder, commonly described as a variational inference network, is responsible for mapping the input to other distributions. The decoder takes the variables and parameters as inputs and projects them to data distributions [57]. To conclude, VAEs are meant to compress the input information into a constrained multivariate latent distribution once VAEs reconstruct/decode to obtain the most exactness in the results possible.

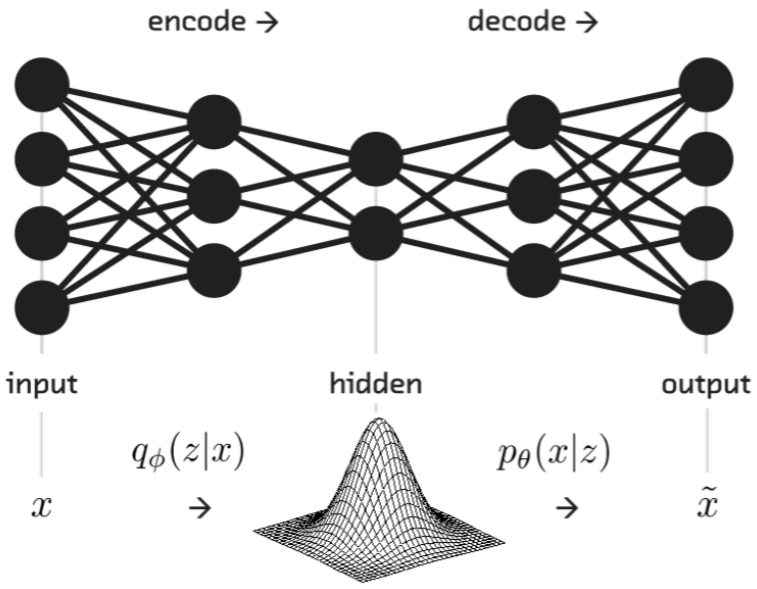


Figure 16: Typical Structure of Variational Autoencoders. Source [57].

Generative Adversarial Networks GANs are a deep generative model. Such as VAEs, GANs algorithms used in unsupervised machine-learning problems. GANs are composed of two NN, a generative and

a discriminative neural network [57]. The generative and discriminative neural networks complete each other to achieve balance in training. Figure 17 illustrates the GANs architecture. Usually, the generative NN is liable for removing noise as input and rendering samples. On the other hand, the discriminative NN is responsible for evaluating and distinguishing the generated samples from training data. Like VAEs, generative networks conduce latent variables and parameters to data distributions. With a training set, this technique can learn how to generate new data with the same statistics as the training set. To conclude is important to point out that as a form of a generative model for unsupervised learning, GANs also have proved to be useful for supervised and reinforcement learning tasks [56].

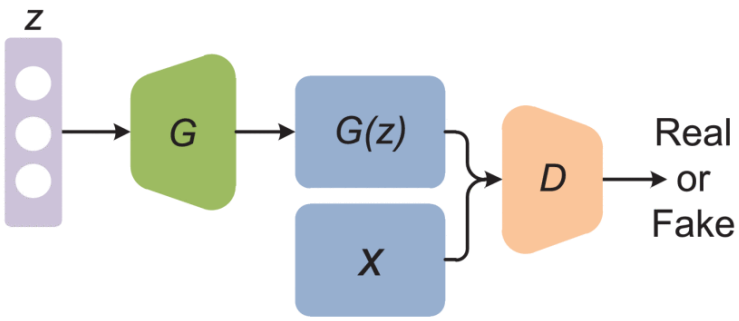


Figure 17: Architecture of Generative Adversarial Networks. Source [56].

3.4.6 Object Counting with Deep Learning Algorithms

The big goal of this dissertation is to perform automatic counts of cells, namely microglial cells. The quantification of cells can be done in two different ways, manually and automatically. Regarding the last one, two methodologies are associated with it, a more classical and a deep learning-based one. The objective of this subsection is to review, within all relevant work and literature, different deep learning-based approaches to the automatic quantification of cells problem. In total, two articles will be brought for discussion to present some aspects regarding how to do object counting with deep learning algorithms. Some of those will be taken into consideration when designing a final solution.

Automatic Ground Truth for Deep Learning Stereology of Immunostained Neurons and Microglia in Mouse Neocortex

The article produced by Phoulady et al. [58] consists of an automatic segmentation algorithm to apply to stereology counts. Their validation experiments were performed in microglial cells containing the IBA1 marker. The microglia were automatically counted in tissue sections of a mouse neocortex. In addition to their objective, the results obtained were compared with manual counts, to prove the benefits of a deep learning-based approach to the cell quantification problem.

Introduction In the referred study, the author's explored automatic segmentation algorithms for automatic stereology counts. Their work delivers promising results for auto-segmentation datasets to train models of CNNs to do automated stereological brain cell counts. In addition to the objective of segmentation and automatic stereology counts, they were also able to compare the results with the manual counting process, to defend that this automatic quantification system can replace manual systems.

Experimental Work In their development, they started using the ASA to automatically count neurons. As the neurons have different characteristics, different size and appearance, they quickly realized that to perform microglial cell counts they would have to make changes to the proposed method. So, algorithm parameters were modified to achieve acceptable results. They changed the minimum clump size, as microglia cells are smaller compared to neurons. They changed the structuring element used in morphological operations to enlarge the radius and the offset. With these changes was possible to quantify accurately and effectively microglial cells with IBA1 marker.

Results In terms of results, they initially designed their approach for automatic counts of NeuN-stained neurons. They identified that to assess the adaptability of their approach, they needed to implement minor modifications to their CNNs algorithm to detect and count IBA1-immunostained microglia cells. They were able to validate their results by checking parameters and testing on the remaining mice samples. In this way, they were able to count microglial cells faster and more efficiently.

Discussion and Conclusions To sum up, this paper provided a general overview of an automatic segmentation algorithm for automatic stereology counts. Their validation experiments used microglial cells containing the IBA1 marker. The results obtained were compared to manual counting, and they managed to defend that this automatic quantification system can replace manual systems, as they are less labour-intensive and less tedious. They have also proven that they can decrease the time spent on counting and decrease efforts and costs while increasing the accuracy, precision and throughput of the final results. This article was also able to verify that by using an automatic segmentation algorithm in conjunction with deep learning models, we can automatically count images of microglia more than five times faster than manual stereology and counting systems.

An Adaptive Digital Stain Separation Method for Deep Learning-based Automatic Cell Profile Counts

In Dave et al. [59] a method to separate stain colour channels on images from tissue sections was developed. The samples of microglial cells used contained the IBA1 marker. Their methodology can function as an input for deep learning-based algorithms to automatically quantify cells. They were able to compare their results against the state-of-the-art methods and conclude that their method manages to overcome them and obtain more accurate results.

Introduction Through this study, automatic deep neural network-based methods have been studied. Combined with hand-crafted techniques, such as image processing and analysis, ML algorithms enable automated and semi-automated counts with high levels of accuracy, precision and efficiency compared to manual counting methods. For that, their segmentation algorithm automatically generates masks on grey-scale images to train CNNs to make automatic counts on images. Their study provides a new proof-of-concept for deep learning-based automatic counting of cells.

Experimental Work In terms of developed work, the authors provide a complete overview of the automatic counting system developed. The stain separation method was based on SNMF for colour-to-grey transformation and its application on 3D stacks. Their approach is to represent 3D information distributed among a stack of images in a 2D image and a post-processing step to match such 2D images of the counterstain dataset. During stain separation, an objective function of SNMF was estimated by applying unsupervised learning on each image. Their purpose was to minimize the cost of the complete image.

Results The results of automatic counting and the model performance were validated based on unseen images not used for training the model. Since 1 of each of the 5 animals is left out for testing the folds, in their approach they were able to cross-validate the performance of the deep learning model on previously unseen images. 3-fold cross-validation was performed on the dataset. With this, and through the results obtained, it was possible to attest that the suggested method overwhelms the limits of the state-of-the-art methods.

Discussion and Conclusions The authors were able to implement an automatic segmentation algorithm for automatic stereology counts combined with an adaptive method to separate stain colour channels on images from tissue sections to expand the automatic counting methods. They proved that automatic deep neural network-based methods have the potential to do more efficient counts on brain cells when compared to state-of-the-art methodologies. To conclude, they evidenced an increased value of accuracy of automatic approaches when compared to the manual counting process.

3.5 Summary

Image processing is a method to convert an image to a digital aspect, allowing users to perform actions to get an enhanced image or extract useful information. This technique is used to improve raw images. Therefore, distributed into several groups emerge "Visualization", "Image Retrieval", and "DIP". The DIP technique consists of the processing of images done by computers. First, images are converted into a digital form (transformed into a computerized structure), and then further preparation is done on those images. There are several techniques of DIP. However, only four are related to the automatic cell counting process using classical and deep learning methodologies. These techniques are "Image Segmentation", "Classification", "Image Restoration", and "Image Enhancement".

Image segmentation allows partitioning an image into several regions, several subparts, and sometimes even divides the image into pixels to help in the analysis of substances, borders and other records relevant to processing. The outcome is a set of sections that together cover the total image. The main goal of segmentation is to simplify a raw image in such a manner that is easier to evaluate a complete picture. Classification is the technique used to extract data from images and label images. An appropriate classification scheme and an adequate amount of training samples are the basics for effective classification. There are various classification approaches such as ANNs and Fuzzy Logic. The classification technique is either supervised or unsupervised. Image restoration is the technique through which a corrupted and noisy image is processed in such a manner to construct a perfect image. There are two types of procedures used to reconstruct an image. One technique is to model the image whose quality is degraded. The other technique, known as image enhancement, increases the quality by applying various filters. The image enhancement method modifies components on the images to increase image clarity. This makes it easier to identify key features in images.

Image analysis techniques emulate human vision, including learning and the ability to make a decision based on input. The information is not collected qualitatively, because these systems extract quantitative information from datasets assembled by a set of images. Typically, image analysis techniques are applied to images resulting from image processing techniques. The most commonly used operations associated with image analysis are morphological analysis, measurements, recognition, representation and description. There are several morphological operators to aid in image analysis. Regarding the quantification of the number of cells from brain images, the ones related are "Dilation", "Erosion", "Opening and Closing", "Hit-or-miss Transform", and "Boundary Extraction".

Dilation is the operation that consists of the expansion of an object boundary. The central pixel of the structural element (e.g. nucleus of the cell) cycles through all pixels of the target object. Erosion is the exact opposite of the dilation operation because it causes a contraction of the boundaries of an object. The object is "reduced"according to the shape and size of the structural element. Opening and closing operations are related and intrinsic to dilation and erosion techniques. The output of opening an object is the same as the result of erosion followed by the dilation of an object. The output obtained by closing an object is the same result that we can obtain with dilation followed by erosion of an object. Hit-or-miss transform involves several basic operations such as erosion, complement, intersection and difference. The final result includes the coordinates of the object of interest in the image. Boundary extraction is the operation that returns a region of pixels corresponding to the boundary of an object of interest.

Conventional cell counting involves specific sets of tools and devices developed for that purpose. This process is tedious, time-consuming, and inaccurate due to operator-dependent biases. Most of the cell counting processes to this date are manual. However, because cell counting is an important procedure routine that may help in the detection of a serious illness, various study reports are focusing on the experience of the development of new systems. With the automation of cell counting, the process is more time-efficient and has fewer errors.

In the case of the classic approaches for the automatic quantification of cells, software solutions for cell

counting are applicable. The manual standalone cell counting assistants, plug-ins, and guides facilitate cell counting by replacing the manual clicker with multiple digital counters. ImageJ is the platform of choice for image processing and automatic cell counting as it has several tools to approach the problem. The choice of ImageJ for the implementation of an automatic cell quantification system was based a lot on the premise stated before and on the fact that allows studying and designing a solution where concrete and acceptable results are expected. ImageJ is a java-based program for image processing and analysis, inspired by NIH Image, developed at the U.S. National Institutes of Health. Accessible for the public domain, meaning that the source code is openly available and its use is license-free. The software was designed with an open architecture that provides extensibility via java plugins. With these highly varied plugins, we are capable of modifying an existing function or introducing a brand new one due to the fact of being an open architecture software that provides extensibility. Taking into account the vast list of projects that function within ImageJ it is important to focus on the features that distinguish it from others and why it is the software chosen to be the basis for the study of a solution to the problem presented. The most obvious distinction is its simplicity in image processing.

With ImageJ we can do the most basic image processing techniques like background subtraction, brightness and contrast adjustment, image type conversion, smoothing, sharpening, filtering, and binarization. It also contains other features for basic manipulations like geometric transformations such as scaling, zooming, and rotation. It is important to point out that it can display, edit, analyze, process, save and print 8–bit, 16–bit and 32–bit images. Applying image processing tasks is an extremely simple process since it provides an easy-to-understand graphical interface that makes the processing task much easier. To automatically count objects in an image, a set of image processes should be applied to the image so the automatic quantification is more precise. As expected, the results are higher evidence of the cells, as was increased the gap between the background and cell pixels. Before automatic counting, thresholding is a fundamental procedure. A manual adjustment with the sliders may be required to decrease the overlap, but is also possible to rely on the automatic adjustment provided by ImageJ. The result is an image in which the white areas have no interest in quantification.

In the case of the DL approaches for the automatic quantification of microglial cells, the problem can be categorized as detection-based counting and regression-based counting. The first approach requires the detection or segmentation of every cell before counting, which implies a supervised learning process. To convert a counting task into a segmentation task, cell annotation is needed, to train the detection or segmentation model. The regression-based cell counting avoids the challenging task of detection or segmentation of single cells because they generate cell density or cell count directly from the images.

To better understand the capabilities of this approach, certain basic notions must be addressed, such as ML and DL. ML is a sub-field of AI. In a nutshell, we can define it as the evolving branch of computational algorithms designed to emulate human intelligence by learning from the surrounding environment. Through various techniques, ML models and algorithms can process large amounts of data and extract useful information. As expected, they can improve upon their previous iterations. ML models and algorithms today can outperform the rival state-of-the-art algorithms and human performance. The obtain

better results learning paradigms are being applied to train ML algorithms. Currently, ML is being successfully applied in diverse fields rating from pattern recognition, computer vision, finance, entertainment, and computational biology to biomedical and medical applications. DL is a sub-discipline of ML, which itself makes it a sub-field of AI. The main distinguishing factor between ML and DL is that we consider the latter to be more complex. It usually takes human interaction to label the data and make it readable by the program, with ML algorithms dealing with structured, labelled data. However, when we resort to DL algorithms, they can process this data accurately without the need for human labelling.

There are some DL theoretical foundations essential to point out like artificial neurons, ANNs and CNNs. The neuron represents the basic computational unit of the brain. Usually, neuron inputs come from dendrites. The artificial neurons are the base element of ANNs. They transform a set of inputs into a single value using a weighted sum since each input has a respective weight. The neuron output values are calculated by applying an activation function. The term ANNs is usually used to characterize the mathematical model composed of a collection of artificial neurons. These neurons are interlinked in a network by layers to learn more complex data relations. FNNs and RNNs stand as two types of ANNs. The FNNs are characterized as being an acyclic graph where the data only can move forward. Unlike these, with RNNs, the data/information can flow in any direction, including to the same layer. RNNs are an extended representation of FNNs with the addition of feedback connections. CNNs follow the same principle as they derive from ANNs. However, it is important to point out that they differ in one simple detail, as they assume that the input is a set of images. The architecture of CNNs disposes of the neurons differently when compared to ANNs as they look to improve the creation of the model. Neurons are disposed of in three dimensions such as width, height and depth. The idea behind CNNs is to use small ANNs convoluted along with the image to extract relevant information. It is important to point out that this idea differs from a fully connected architecture which demands much more parameters.

To improve the accuracy of predictive models ML techniques are required. Depending on the nature of the problem, the ML paradigms vary. There are different approaches based on the type and volume of data, which have different amounts and types of supervision in training. Each has its advantages and disadvantages. Supervised learning requires labelled features to define the meaning of the data. The model has a mapping function formed by an algorithm that differs from problem to problem, after being trained to predict the output data for each input. Occasionally, patterns are identified in a subset of the data. Unsupervised learning differs from supervised learning as it can learn from no results. The model tries to learn without any supervision. A machine is provided with just the input to develop a learning pattern. This learning paradigm is best suited to problems that require a massive amount of unlabelled data. Reinforcement learning is a behavioural model. Reinforcement learning uses a method based on rewards and penalties for each action that it takes to train the model. By using this method, this learning paradigm differs from other learning paradigms because it isn't trained with the sample dataset. Therefore, a sequence of successful decisions results in the process being "reinforced".

DL can easily extract information and features that other methods composed of simple representations can't. In some cases, some of these models have limitations because they do not have good

generalizations. To solve this problem, VAEs e GANs can be a better alternative. VAEs provide probabilistic descriptions of observations in latent spaces. VAEs have ANNs architecture. Therefore, they belong to the families of probabilistic graphical models and variational bayesian methods. Unlike more common approaches, such as NN as regressors or classifiers, VAEs are a powerful generative model. GANs are a deep generative model. Such as VAEs, GANs algorithms used in unsupervised machine-learning problems. GANs are composed of two neural networks, a generative and a discriminative neural network. The generative and discriminative neural networks complete each other to achieve balance in training. The generative NN is liable for removing noise as input and rendering samples, and the discriminative NN is responsible for evaluating and distinguishing the generated samples from training data.

4

Data

This chapter presents the reader with all the relevant information about the used data. First, we have a description of a generic cell sample. This cell sample had the purpose of being used to fully understand ImageJ limitations, as well as study what would be the best approach to the problem based on a more classical methodology. With the same cell sample, a deep learning-based approach was also studied. Next, we detail all the information related to the dataset of microglial cells (brain images), ranging from the data overview to its lobule and deep cerebellar nuclei segmentation. Finally, the manual quantification of cells is presented since it will be used to compare the performance of all methodologies.

4.1 Cell Colony Sample

According to the study of relevant literature and documentation about the automatic quantification of cells problem, more specifically related to the classic approach detailed in section 3.3, the software of choice is ImageJ. To better understand the extensivity of ImageJ, a generic cell sample was selected to do some experiments. In section 5.2.1, these experiments are detailed. Regarding a deep learning-based approach, and considering all the conclusions brought from the study of all relevant literature and documentation carried out in section 3.4, the best approach to the problem is the use of CNNs. Therefore, the same cell sample was used to study the best strategy for this deep learning model. However, unlike the approach implemented with the more classical methodology, in this case, the cell sample was fractionated into 16 parts to build a better deep learning model. The experiments carried out are detailed in section 5.3.1. Next, follows explicit information about this cell sample.

As stated, the cell colony sample was used in the first place to understand the possible limitations of ImageJ and better define a work strategy. Available at `https://imagej.net/images/Cell_Colony.jpg`, belonging to the ImageJ documentation and intended for public use, this cell colony sample was selected. Figure 18 illustrates the cell colony sample, and Table 1 contains more detailed information about this sample. The image processing and analysis techniques applied to enhance some of the image characteristics, allowing to more easily identify the cells for later counting are detailed in subsection 5.2.1, which is related to the experiments carried out.

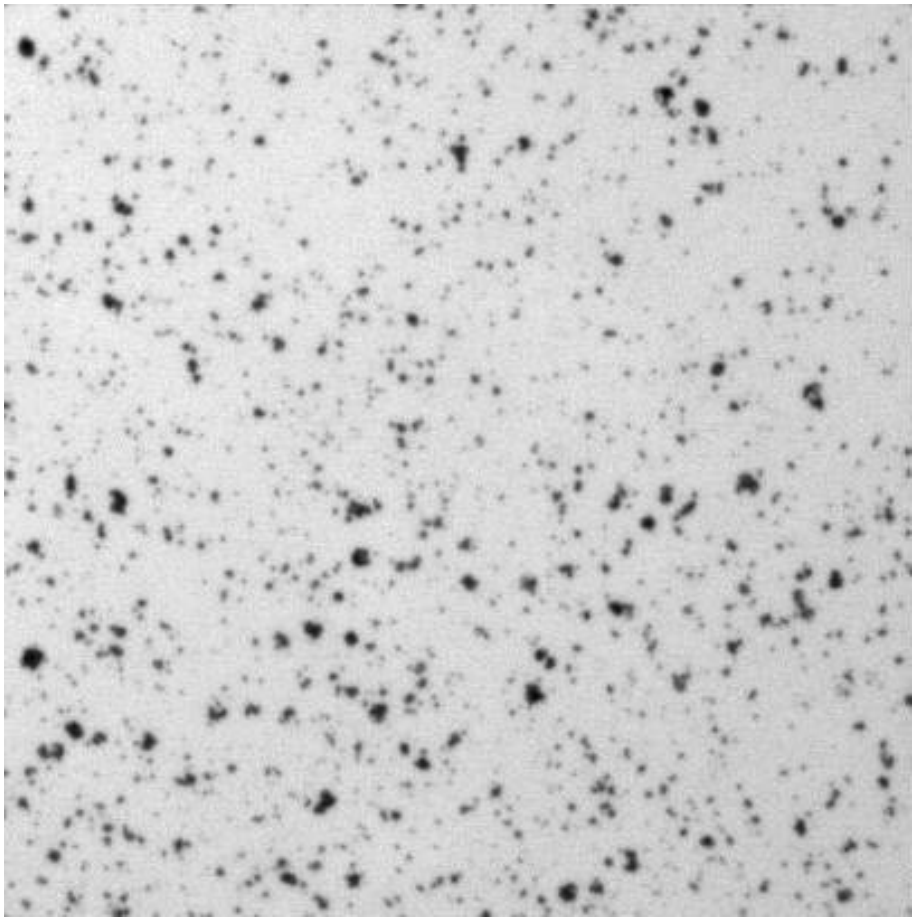


Figure 18: Cell Colony Sample.

Size	Width	Height	Pixel Size	Bits per Pixel	Display Range
162K	406 pixels	408 pixels	1x1 pixel^2	8-bit	0-255

Table 1: Cell Colony Sample Information.

Figure 19 illustrates the cell sample with the particularity of being divided into 16 parts. The rationale for this division is related to the fact that it is necessary to build a dataset to test the best strategy and possible limitations of CNNs models to quantify cells. Table 2 contains detailed information about each section resulting from the division made. The experiments implemented are detailed in subsection 5.3.1.

Size	Width	Height	Pixel Size	Bits per Pixel	Display Range
10K	101 pixels	102 pixels	1x1 pixel^2	8-bit	0-255

Table 2: Cell Colony Sample (Partitioned) Information.

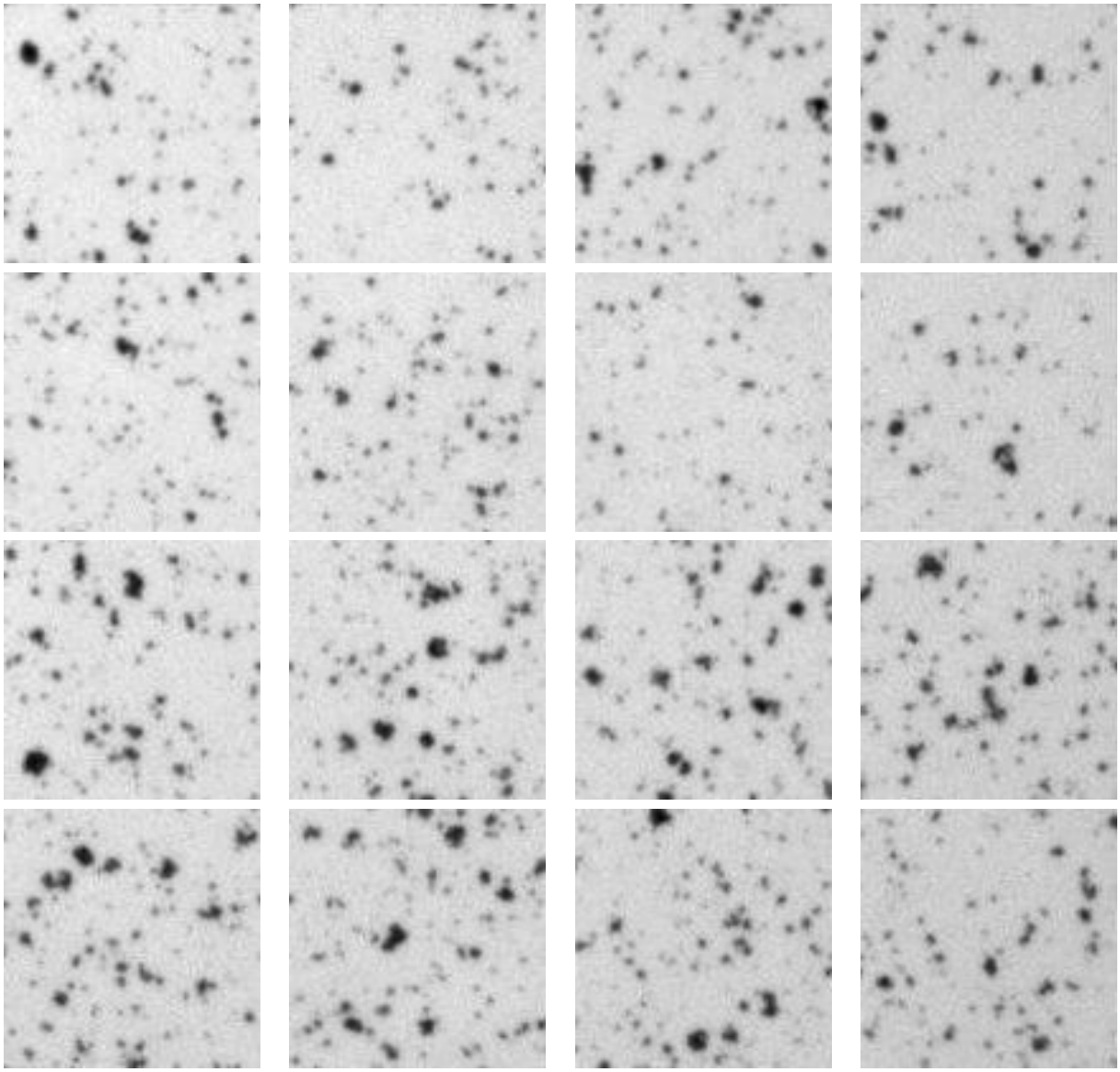


Figure 19: Cell Colony Sample (Partitioned).

4.2 Brain Image Samples

This dissertation aims to study the advantages of two different methodologies, a classical and a deep learning-based approach, to automatically quantify microglial cells. In the end, it is expected that one of these methodologies will be a more reliable solution than the current cell counting processes, which are usually performed manually. For this, it was necessary to resort to brain images of four different animals since they will originate the study and the implementations carried out in both approaches.

As stated before, microglia are a type of neuronal cell located throughout the brain and spinal cord. Taking this into consideration, the only non-evasive way to access them is through neuroimaging. These brain images contain a large population of microglia that are in a reactive shape. They contain the IBA1 marker, which is a marker upregulated in reactive microglia and used to visualize these cells. The data

acquisition and the manual quantification of cells were done together and in partnership with the School of Medicine of the University of Minho. The next subsections present and detail all the relevant information related to these images, ranging from data acquisition and overview to the lobule and Deep Cerebellar Nuclei (DCN) segmentation process. The last subsection presents the results of the manual counting process, which with all the conclusions drawn, will help to cease the best solution for the automatic quantification of microglial cells.

4.2.1 Data Acquisition & Overview

The dataset is composed of 16 brain images from 4 different animals. To access the microglia density and morphology, 4 coronal brain sections per animal ($n = 4$ per genotype) were imaged twice (in both hemispheres) for each region of interest (DCN and Cervical Spinal Cord (CSC)) to yield 4–6 digital photomicrographs per section. For the Pontine Nuclei (PN), 4 sagittal brain sections per animal were used ($n = 3$ animals for wild-type and $n = 4$ animals for CMVMJD135), and 2 photomicrographs per section were taken [60]. To stack all images the Olympus Confocal FV1000 laser scanning microscope with a resolution of 1024 × 1024 px using a 40× objective (UPlanSApo, N.A. 0.90; dry; field size 624.39 × 624.39 μm; 0.31 μm/px) was used. The acquisition settings for the images were the following: scanning speed = 4 μm/px; pinhole aperture = 110 μm; Iba-1, excitation = 559 nm, emission = 618 nm; in a 3-dimensional scenario (X, Y, and Z axes). In addition, the ImageJ was used on Z-stacked 3D volume images from sections of the affected brain regions (DCN, CSC, and PN) [60].

4.2.2 Lobule and Deep Cerebellar Nuclei Segmentation

The DCN consist of three nuclei: the fastigial nucleus, the interposed nucleus and the dentate nucleus. Together they form the output of the cerebellum. The fastigial nucleus is the most medially located of the cerebellar nuclei, as they receive input from the vermis [61]. The lobules of the cerebellum are the smallest of the lobes of the flocculonodular lobe. This flattened lobe lies between the posterolateral fissure (inferiorly), the inferior medullary velum, and the cerebellar peduncles (superiorly). That said, to better quantify microglial cells, in this study, the brain images of the animals were separated in several areas. One of these areas is the DCN and the remaining ones are the lobules. As stated, each brain image contains a DCN area, where, as a rule, the number of cells there is higher when compared to the number of cells in the lobules. The remaining areas of the image are composed of several lobules, where we have microglial cells more dispersed throughout the lobe in question. Figure 20 illustrates two examples of the lobule and deep cerebellar nuclei segmentation performed in this study. The image on the right refers to the segmentation performed in slice 1 of the animal CN282 2TE. The image on the left is evidence of the segmentation performed in slice 2 of the same animal. In subsection 4.2.3 the results obtained from the manual quantification of microglial cells are presented, differentiating between the counts of the DCN and lobule areas. The number of lobules that compose each brain image is given.

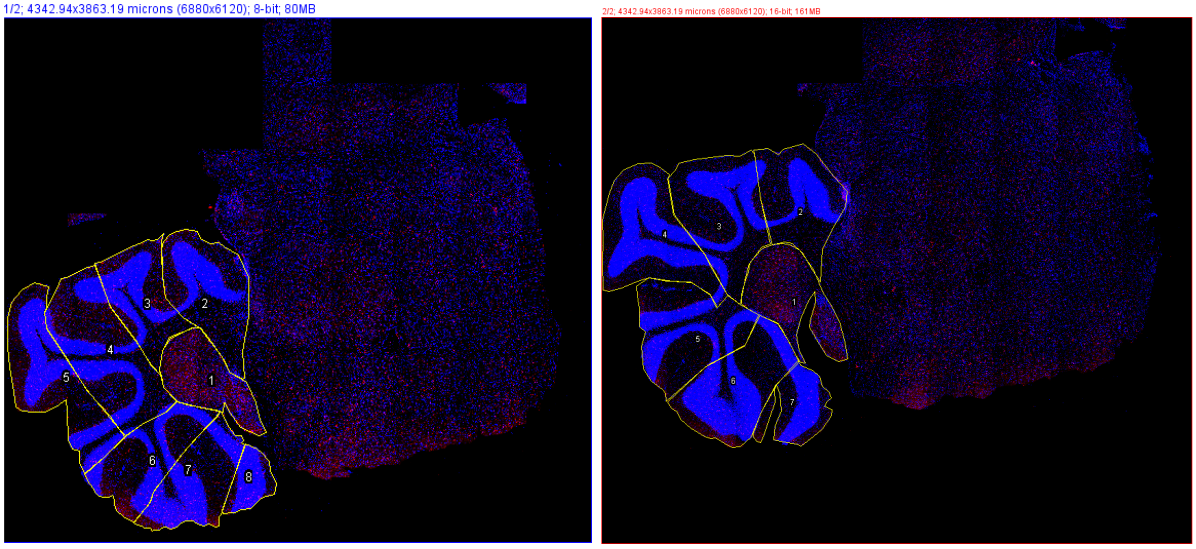


Figure 20: Lobule and Deep Cerebellar Nuclei Segmentation Representation.

4.2.3 Manual Cell Counting

With the help of elements from the School of Medicine, the manual cell counting process was conducted. The total count of IBA1 positive cells was obtained using the multi-point tool. The quantification was performed on images acquired with the acquisition settings described in subsection 4.2.1, normalized first to the total image area and then for volume [60]. The data was obtained from individual cells of the CSC (310 microglial cells from wild-type mice and 389 from CMVMJD135 mice), DCN (349 microglial cells from wild-type mice and 445 from CMVMJD135 mice), and PN (152 microglial cells from wild-type mice and 180 from CMVMJD135 mice). Therefore, the total number of analysed microglia with the IBA1 positive marker was 1825. However, the images that compose the dataset not only contain microglia with the IBA1 positive marker. So, bearing in mind that the scope of this dissertation is to automate the quantification of microglial cells from brain images, the results reveal the complete counting process (IBA1 positive and non-positive IBA1 microglial cells), detailed by the animal, lobule and deep cerebellar nuclei cell count.

CN276 2FD Table 3 presents the results obtained from the counting process and the respective area of each slice. In total, the DCN area of the animal CN276 2FD contains 668 microglial cells. Table 4 details the results brought from the quantification process of slices 1 and 2. Slice 1 contains ten lobules. The number of microglial cells is 842. On the other end, slice 2 contains eighteen lobules with a total number of 1022 microglial cells. Table 5 shows the results fetched from the quantification process of the lobules of slices 3 and 4. Slice 3 contains eleven lobules, and slice 4 includes twelve lobules. The number of microglial cells in the lobules of slice 3 is 863, and 1274 in slice 4.

Slice	Number of Cells	Area (μm)
1	147	288932.803
2	163	285064.506
3	205	363675.236
4	153	277021.494

Table 3: CN276 2FD - Deep Cerebellar Nuclei Cell Count.

Slice	Region	Number of Cells	Area (μm)
1	Lobule 2	73	197029.686
	Lobule 3	80	233549.782
	Lobule 4	54	221957.245
	Lobule 5	38	198076.453
	Lobule 6	43	181218.603
	Lobule 7	52	166547.527
	Lobule 8	116	285983.765
	Lobule 9	186	520392.636
	Lobule 10	105	426406.744
	Lobule 11	95	280652.704
2	Lobule 2	61	212812.877
	Lobule 3	47	142765.554
	Lobule 4	39	127533.440
	Lobule 5	41	125675.798
	Lobule 6	43	142936.496
	Lobule 7	47	133967.850
	Lobule 8	61	226804.968
	Lobule 9	74	279216.637
	Lobule 10	56	173600.354
	Lobule 11	97	315884.563
	Lobule 12	64	222045.704
	Lobule 13	51	149114.692
	Lobule 14	50	197360.412
	Lobule 15	32	157927.140
	Lobule 16	33	120969.131
	Lobule 17	65	265616.635
	Lobule 18	87	313512.104
	Lobule 19	74	178078.700

Table 4: CN276 2FD - Slice 1 & 2 - Lobule Cell Count.

Slice	Region	Number of Cells	Area (μm)
3	Lobule 2	115	367417.617
	Lobule 3	116	381444.375
	Lobule 4	45	174852.729
	Lobule 5	33	136708.092
	Lobule 6	60	257572.427
	Lobule 7	115	423266.443
	Lobule 8	90	281917.033
	Lobule 9	90	259198.652
	Lobule 10	85	291940.415
	Lobule 11	72	209955.486
	Lobule 12	42	53199.434
4	Lobule 2	68	196495.743
	Lobule 3	70	195813.571
	Lobule 4	65	188390.571
	Lobule 5	153	455472.759
	Lobule 6	131	308181.840
	Lobule 7	128	401349.684
	Lobule 8	119	336006.238
	Lobule 9	140	461858.954
	Lobule 10	130	353812.435
	Lobule 11	109	306318.619
	Lobule 12	105	224807.862
	Lobule 13	56	142629.678

Table 5: CN276 2FD - Slice 3 & 4 - Lobule Cell Count.

CN282 2TE Table 6 presents the results obtained from the counting process and the respective area of each slice. The DCN area of the animal CN282 2TE contains 747 microglial cells. Table 7 points to the results brought from the quantification process of slices 1 and 2. Slice 1 contains seven lobules with 1056 microglial cells. On the other end, slice 2 contains six lobules with a total number of 780 microglial cells. Table 8 displays the results from the quantification process of the lobules of slices 3 and 4. Slices 3 and 4 both contain six lobules, with 843 and 854 microglial cells, respectively.

Slice	Number of Cells	Area (μm)
1	162	307663.437
2	198	350889.696
3	171	350064.077
4	216	329267.003

Table 6: CN282 2TE - Deep Cerebellar Nuclei Cell Count.

Slice	Region	Number of Cells	Area (μm)
1	Lobule 2	215	441463.932
	Lobule 3	126	335660.769
	Lobule 4	187	575328.978
	Lobule 5	166	482161.533
	Lobule 6	161	375729.593
	Lobule 7	142	360554.061
	Lobule 8	59	116713.527
2	Lobule 2	148	401681.207
	Lobule 3	169	503307.263
	Lobule 4	156	528835.309
	Lobule 5	119	487064.642
	Lobule 6	136	457288.653
	Lobule 7	52	205100.590

Table 7: CN282 2TE - Slice 1 & 2 - Lobule Cell Count.

Slice	Region	Number of Cells	Area (μm)
3	Lobule 2	205	488441.338
	Lobule 3	128	413963.486
	Lobule 4	171	618159.959
	Lobule 5	151	398723.004
	Lobule 6	98	322174.330
	Lobule 7	90	337119.151
4	Lobule 2	216	562527.500
	Lobule 3	145	400294.549
	Lobule 4	149	474527.346
	Lobule 5	94	365960.431
	Lobule 6	158	425694.289
	Lobule 7	92	365867.987

Table 8: CN282 2TE - Slice 3 & 4 - Lobule Cell Count.

CN283 2FD Table 9 presents the results obtained from the counting process and the respective area of each slice. In total, the DCN area of the animal CN283 2FD contains 597 microglial cells. Table 10 details the results brought from the quantification process of slices 1 and 2. Slice 1 contains six lobules. The number of microglial cells is 743. On the other end, slice 2 contains seven lobules with a total number of 921 microglial cells. Table 11 shows the results fetched from the quantification process of the lobules of slices 3 and 4. Slices 3 and 4 both contain seven lobules. The number of microglial cells in the lobules of slice 3 is 658, and 825 in slice 4.

Slice	Number of Cells	Area (μm)
1	151	437991.311
2	156	321837.627
3	121	305943.663
4	169	393448.924

Table 9: CN283 2FD - Deep Cerebellar Nuclei Cell Count.

Slice	Region	Number of Cells	Area (μm)
1	Lobule 2	118	707415.277
	Lobule 3	133	526210.222
	Lobule 4	107	422756.010
	Lobule 5	129	515264.394
	Lobule 6	104	434819.133
	Lobule 7	152	602430.561
2	Lobule 2	133	445827.122
	Lobule 3	115	360652.880
	Lobule 4	163	454541.547
	Lobule 5	146	448844.297
	Lobule 6	159	498722.925
	Lobule 7	113	343325.240
	Lobule 8	92	317199.098

Table 10: CN283 2FD - Slice 1 &2 - Lobule Cell Count.

Slice	Region	Number of Cells	Area (μm)
3	Lobule 2	81	275852.000
	Lobule 3	93	422877.143
	Lobule 4	130	670650.923
	Lobule 5	67	343893.052
	Lobule 6	100	450314.632
	Lobule 7	114	511644.342
	Lobule 8	73	318657.878
4	Lobule 2	101	333694.346
	Lobule 3	121	357793.497
	Lobule 4	153	536827.715
	Lobule 5	170	499677.248
	Lobule 6	133	478029.851
	Lobule 7	82	309468.483
	Lobule 8	65	134080.616

Table 11: CN283 2FD - Slice 3 & 4 - Lobule Cell Count.

CN284 TDTE Table 12 presents the results obtained from the counting process and the respective area of each slice. The DCN area of the animal CN284 TDTE contains 619 microglial cells. Table 13 points to the results brought from the quantification process of slices 1 and 2. Slice 1 contains six lobules with 637 microglial cells. On the other end, slice 2 contains nine lobules with a total number of 831 microglial cells. Table 14 displays the results from the quantification process of the lobules of slices 3 and 4. Slice 3 contains five lobules, and slice 4 includes six lobules. The number of microglial cells in the lobules of slice 3 is 430, and 716 in slice 4.

Slice	Number of Cells	Area (μm)
1	151	362609.342
2	211	406624.162
3	119	293392.022
4	138	433481.885

Table 12: CN284 TDTE - Deep Cerebellar Nuclei Cell Count.

Slice	Region	Number of Cells	Area (μm)
1	Lobule 2	124	408437.576
	Lobule 3	85	330030.462
	Lobule 4	136	445978.538
	Lobule 5	57	327983.548
	Lobule 6	115	525734.455
	Lobule 7	120	526253.655
2	Lobule 2	104	300994.332
	Lobule 3	99	339689.249
	Lobule 4	87	361807.631
	Lobule 5	168	830300.227
	Lobule 6	72	345262.177
	Lobule 7	55	335529.675
	Lobule 8	64	221446.015
	Lobule 9	92	408294.527
	Lobule 10	90	176739.061

Table 13: CN284 TDTE - Slice 1 & 2 - Lobule Cell Count.

Slice	Region	Number of Cells	Area (μm)
3	Lobule 2	93	324649.992
	Lobule 3	142	540196.337
	Lobule 4	75	669286.978
	Lobule 5	63	400084.160
	Lobule 6	57	280211.205
4	Lobule 2	130	629810.273
	Lobule 3	169	690169.322
	Lobule 4	136	401736.195
	Lobule 5	99	430521.292
	Lobule 6	106	466163.968
	Lobule 7	76	518805.950

Table 14: CN284 TDTE - Slice 3 & 4 - Lobule Cell Count.

5

Experimental Setup

Having presented all the main theoretical foundations and relevant work concerning the automatic quantification of cells, we now move on to the design and conception of solutions that aim to solve the problem. Therefore, this chapter describes to the reader all the experiments performed during this dissertation work. We start with a description of both experimental environments used in our approaches. Then the experiments conducted with some generic cell samples are presented. Following that, and with the set of brain images, the experiments carried out for automatic quantification of microglial cells, using a classical and deep learning methodology, are described.

5.1 Experimental Environments Description

Throughout all the work performed during this dissertation, it was necessary to resort to two experimental environments. The reason behind these different environments stands to improve performance and yield better results. That said, the first of those environments is the local one, where all the experiments related to the so-called classical approaches were conducted. Additionally, the first test fits made to the selected deep learning model, which automatically quantifies cells, were also implemented in the local machine. However, through the results obtained and the execution times of the model, it was not difficult to realize that performing an implementation in the local environment was not the best option. Thus, it was necessary to use a cloud environment, namely Google Colab Pro, to carry out all the experiments related to deep learning approaches. In the following subsections, we find more detailed information about these environments. Subsection 5.1.1 presents all information related to the local environment, and subsection 5.1.2 presents all information related to the cloud environment.

5.1.1 Local Environment

All the experiments on the classic methods were executed on the same machine so that the performance obtained in each of the tests is not affected by the fact that they were conducted in environments with different specifications. Additionally, as previously stated, the first test fits made to the selected deep

learning model were also implemented in the local machine. Table 15 presents the machine specifications. The given information was gathered from Intel Support and ASUS Customer Support.

ASUS GL552VX	
Processor	Intel® Core™ i7-6700HQ
Processor's Base Frequency	2.60 GHz
Max Turbo Frequency	3.50 GHz
# Cores	4
# Threads	8
Cache Memory	6 MB Intel® Smart Cache
RAM Memory	16 GB DDR4 2400 MHz
Storage	1 TB HHD + 256 GB SSD
GPU	Nvidia ® GTX 950m 4GB

Table 15: Experimental Local Environment Specification.

5.1.2 Cloud Environment

All additional experiments and the results gathered with a deep learning-based methodology were executed on the same machine (cloud environment) so that the performance obtained is not affected. Initially, test fits made to the selected deep learning model were implemented in the local machine. However, as Google Colaboratory is probably the most popular hosted Jupyter notebook service in the world, it was easy to understand that for more acceptable results and better execution times, we would have to resort to this cloud environment. Given the size of our dataset, a total of six hundred sixty-one images of microglial cells, and knowing that the free version of Google Colaboratory does not allow for too long execution times, it was necessary to resort to the Pro version. Table 16 presents the specifications of Google Colaboratory Pro. The given information was gathered from Google Colab documentation.

Google Colab Pro	
GPU	Nvidia ® K80, P100, T4
GPU Memory	16 GB
GPU Memory Clock	0.82 GHz / 1.59 GHz
RAM	32 GB
CPU	2 x vCPU
Performance	4.1 TFLOPS / 8.1 TFLOPS

Table 16: Experimental Cloud Environment Specification.

5.2 Classic Methods for Automatic Cell Counting

The software chosen to develop a solution for automatic cell counting based on a more classic approach was ImageJ. As stated in subsection 3.3.1, this is a public domain program that is open-source

and license-free. This software runs on any computer with a Java 1.5 or later virtual machine. In order not to affect and influence the results obtained in both approaches studied, within the classical methodology, the ImageJ version was the same in both cases. Figure 21 illustrates the version of ImageJ (version 1.53k) used to carry out the work and the version of the Java virtual machine running.

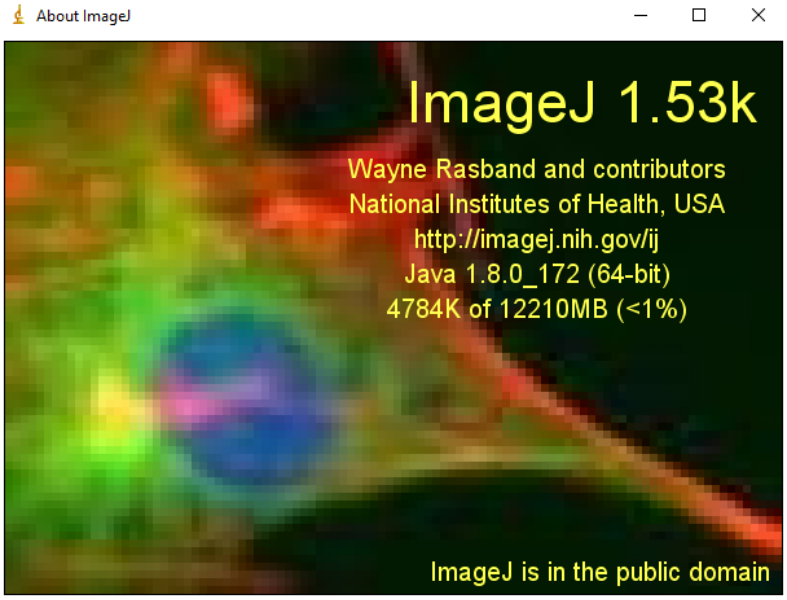


Figure 21: ImageJ Version.

According to the study carried out, the most suitable solutions to apply to this work and automatically count cells with ImageJ are the Analyze Particles functionality and the Image-based Tool for Counting Nuclei (ITCN) plugin. These two were selected as they have proven to be successfully applied in similar cases. In the end, through the analysis of the results and the entire counting process, the objective is to substantiate their pros and cons, and which one can better answer the problem of cell counting within the classical approaches.

In a nutshell, the Analyze Particules functionality consists of a set of commands and steps to count and measure objects in binary or thresholded images. This functionality scans the image or the selected area until it finds the edge of an object. Is necessary to configure the particle analyzer, as particles outside the range specified are ignored. Particles with circularity values outside the range specified are also ignored. In the end, the object is outlined and measured and the results are displayed. This functionality also enables it to work with RGB images. The results are calculated using brightness values and RGB pixels are converted to brightness values. In some cases, other types of image processing are needed, such as work with outlines which this functionality allows. In subsection 5.2.1 all the experiments conducted with the Analyze Particles functionality and the respective cell counting process implemented for the cell colony sample are described. In subsection 5.2.2 using the developed Analyze Particles protocol, the entire counting process which led to the final results, is explained step by step.

Developed by the Center for Bio-image Informatics the ITCN is an ImageJ plugin that automatically

quantifies the number of cells in an image. This plugin's workflow requires an estimation of the diameter of a cell, which has to be done by the user, an estimation of the minimum distance between cells, also done by the user, and either a region of interest (ROI) selection or a black and white mask image that is white in regions that are to be counted. The following subsections detail all the experiments conducted with the cell colony sample and brain images with the ITCN plugin. In subsection 5.2.1 all the experiments conducted with the ITCN plugin and the respective cell counting process implemented for the cell colony sample are described. In subsection 5.2.2 also with the ITCN plugin, the quantification process is explained.

5.2.1 Cell Colony Sample

A particular feature of ImageJ is the possibility of counting objects. These objects can be cells that are in 2D images. Thus, ImageJ fits perfectly into the context of this dissertation since it allows for solving the presented problem. As previously stated, to count objects with ImageJ, following a set of techniques is mandated. Alternatively, a certain plugin can replace the obligation of following a set of pre-established procedures. The Analyze Particles functionality and the ITCN plugin were chosen for the automatic quantification of cells. To understand the extensiveness of these two different strategies, the generic cell sample, detailed in section 4.1, was selected to carry out some experiments. This helped to fully comprehend the best approach to automatically count cells.

Analyze Particles

The Analyze Particles consists of a set of stages that lead to automatic counting. To count objects, which in this case are the cells present in the cell colony image, a set of processes should be applied to make the count more reliable. Background subtraction is one of these steps as it results in higher evidence of the objects of interest, therefore, increasing the gap between background pixels and objects. In the different experiments, the subtraction of background was tested to verify its benefits. Thresholding the image is also a fundamental procedure. This process specifies the image binarization and what must be included in the analysis based on the intensity of the pixels. ImageJ offers the possibility to adjust the threshold value manually or automatically. The manual adjustment with sliders requires decreasing the overlap as much as possible and is more trustful when compared to the automatic adjustment. In the experiments conducted, several values for the threshold were tested. However, it is important to notice that in the case of this generic cell sample, all areas of interest are included with or without overlap. After removing the background and defining the threshold value follows the automatic quantification. The control menu window displayed in Figure 22, referring to one of the counts performed in the experiments, allows the specification of the range of the size of objects that need to be counted. This control menu also entitles to set which objects are included based on the cell circularity, where the minimum extent means that the object corresponds to a linear line and the maximum corresponds to a flawless circle. The result of this process is an 8-bit image generated from the original image, holding the numbered outlines of the measured objects.

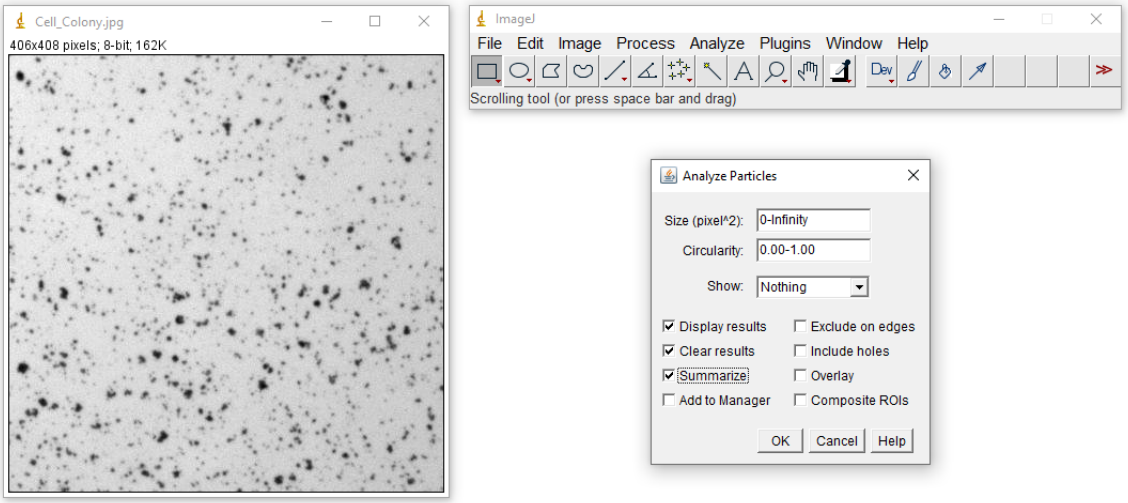


Figure 22: Cell Colony Sample - Analyze Particles.

Moving on to the experiments conducted, Table 17 displays the different settings applied to the images, namely the threshold values applied in each experiment, the range of size and the circularity of the objects that lead to the final results obtained in the counting process. It is important to notice that in all the tests conducted the circularity chosen for the objects range from 0.00 to 1.00, and the range of sizes defined was 0 to infinity since in the cell sample there are cells with different values of circularity and sizes, and in this way, we are not excluding any cell.

		1^{st}	2^{nd}	3^{rd}	4^{th}	5^{th}
Threshold	Min	0	0	129	0	129
	Max	171	202	255	223	255
		6.63%	6.11%	6.09%	4.90%	4.89%
	Algorithm	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	0-Inifinity	0-Inifinity	0-Inifinity	0-Inifinity	0-Inifinity
	Circularity	0.00-1.00	0.00-1.00	0.00-1.00	0.00-1.00	0.00-1.00

Table 17: Cell Colony Sample - Analyze Particles - Quantification Settings.

The 1^{st} experiment is the result of cell counting after applying auto-thresholding to the image, at a pixel value of 171. Figure 23 illustrates the counting process of this experiment. The 2^{nd} test presents the results of the quantification process by applying a background subtraction after the passage of a "rolling ball" with a light background and a radius of 50 pixels. After that, auto-thresholding was applied to the image, at a pixel value of 202. The 3^{rd} experiment is related to the prior procedure, however, after subtracting the background the resulting image was then converted to binary, resulting in an 8-bit image. Then, the watershed algorithm was applied. What this algorithm does is separate with a pixel what it considers to be two or more cells together in the image. Thus, with the application of this algorithm, we can obtain more accurate results in the counting process, since, without the application of this algorithm,

the software only counts one cell when in fact there may be two or more cells. Notice that we are only able to apply this algorithm in binary images. To complete the process, auto-thresholding was applied to the image, at a pixel value of 255. The 4^{th} test is identical to the 2^{nd} test, however, the background subtraction applied with a light background only had 2 pixels in the radius of the "rolling ball". Then again, the auto auto-thresholding was used, at a pixel value of 223. The 5^{th} and final experiment is similar to the previous one. Once the background was subtracted, the resulting image was then converted to binary, an analogous process to the one conducted in the 3^{rd} experience, resulting again in an 8-bit image. Then, the watershed algorithm was applied. To finalise the counting process, auto-thresholding was applied to the image, at a pixel value of 255.

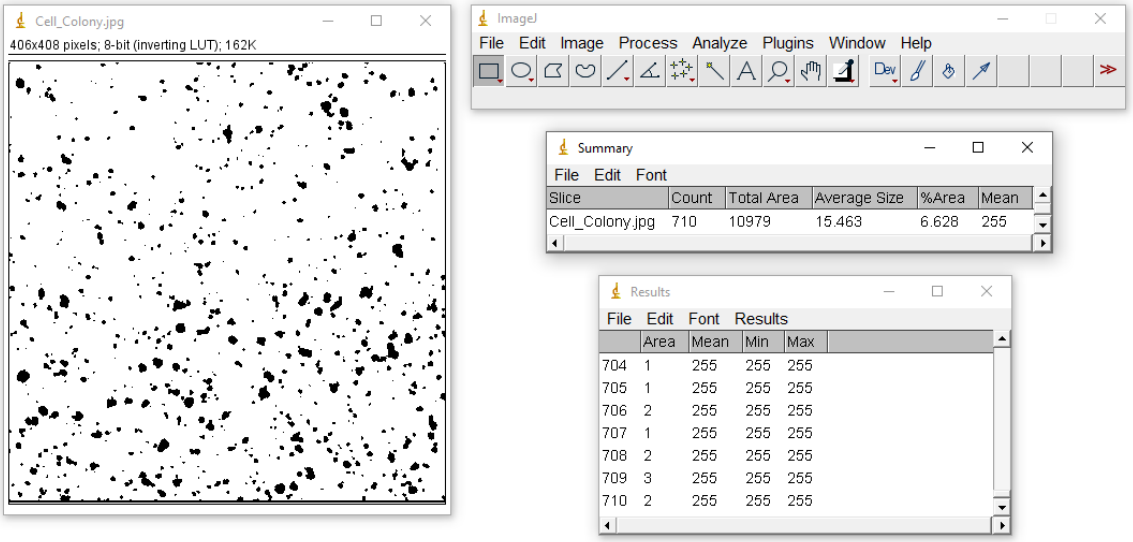


Figure 23: Cell Colony Sample - 1^{st} Experiment - Automatic Quantification.

The results obtained and their respective discussion is presented in subsection 6.1.1. An approach considered optimal for the development of the Analyze Particles protocol, to automatically quantify microglial cells, is also explained based on the results acquired with the experiments conducted with the generic cell sample.

ITCN

As stated, the ITCN is an ImageJ plugin that purpose another approach to automatically count cells in an image. Previous to the quantification of cells, this plugin requires that a set of procedures must be applied to the image to make the count more reliable and effective. Therefore, in 1^{st} experiment the background was subtracted by applying auto-thresholding to the image, at a pixel value of 171. Then, using the ITCN plugin, the size of the cell was set to 9 pixels, by measuring the line that joins the beginning to the end of the selected cell body. The minimum distance between cells was defined by a line that joins the cell whose size was measured to the nearest cell. In this case, the distance was 4 pixels. Notice that in this case, the threshold value defined in the ITCN count was 2.0. Figure 24 illustrates the counting process of

this experiment. In the 2^{nd} test, all the settings implemented in the previous experiment were followed, except for the value defined for the ITCN threshold. So, it will be possible to understand how automatic counting behaves when we change the threshold values. The 3^{rd} experiment is also very similar to the 1^{st}. However, it was possible to sense that the results of the counts are more accurate if we find the smallest cell and the smallest distance between cells. Preserving the values of the ITCN threshold higher is also important. Therefore, keeping the values for the auto-thresholding to remove the background, the cell size was defined in 6 pixels and the minimum distance between cells was 5 pixels. The 4^{th} test is the result of applying a background subtraction after the passage of a "rolling ball"with a light background and a radius of 50 pixels. Then the image was converted to binary to apply the watershed algorithm. In this case, the new cell size was 5 pixels and the distance 5 pixels. The 5^{th} and final experiment had a background subtraction applied only had 2 pixels in the radius of the "rolling ball". Auto-thresholding was used at a pixel value of 223. The cell size was 5 pixels and the distance between cells is 4 pixels.

Figure 24: Cell Colony Sample - 1^{st} Experiment - ITCN.

Table 18 displays all the different settings applied to the images, namely the threshold values, the cell size and minimum distance between cells defined, along with the ITCN threshold value that leads to the final results obtained in the counting process. The results obtained and their respective discussion is also presented in subsection 6.1.1. The pros and cons of implementing an approach to automatically quantify microglial cells with the ITCN plugin are also discussed, based on the results gathered from the experiments conducted with the generic cell sample.

		1^{st}	2^{nd}	3^{rd}	4^{th}	5^{th}
Threshold	Min	0	0	0	129	0
	Max	171	171	171	255	223
		6.63%	6.63%	6.63%	6.09%	4.90%
	Algorithm	Default	Default	Default	Default	Default
ICTN	Cell With	9	9	6	5	5
	Minimum Distance	4	4	5	5	4
	Threshold	2.0	1.0	3.0	4.5	5.0

Table 18: Cell Colony Sample - ITCN - Quantification Settings.

5.2.2 Brain Image Samples

Within the so-called classic approaches emerges ImageJ. As stated, this software offers the possibility of counting objects in 2D images. Thus, ImageJ fits perfectly into the context of this dissertation, namely for the automatic quantification of microglial cells. Collected together with elements from the School of Medicine of the University of Minho, these images are in the desired format for quantification. With image processing and analysis techniques, once again, the Analyze Particles and the ITCN plugin have been chosen. These two different strategies were implemented to the brain images to fully comprehend who is the best approach to automatically count microglial cells regarding a more classical approach. It is important to note that to obtain more accurate results, all DCN areas and Lobules were divided into several images with the same size, to minimize image noise and obtain a clearer view of the cells.

Analyze Particles

Having studied all relevant literature and documentation, namely, the approaches presented in subsection 3.3.3, their respective pros and cons, and the drawn conclusions from the experiments carried out with the generic cell sample, the ImageJ Analyze Particles feature was tested again to automate cell counting. The section where similar work was brought up for discussion helped to support this approach and to design a solution to the presented problem. The different studies fetched good practices that need to be considered, from the beginning of the counting process to the end, such as image processing, treatment and analysis. The experiments carried out with the generic cell sample, detailed in subsection 5.2.1, also brought very relevant considerations to this approach. Above all, analyzing the entire counting process and the inherent results, the Analyze Particles has proven to be a very valid alternative for the automatic quantification of microglial cells.

As previously described, the Analyze Particles consists of a set of procedures before automatic counting. To count objects, a set of processes should be applied to make the count more reliable. The first of those processes was to divide the DCN areas and the Lobules into several images, all with the same size. Figure 25 illustrates the division of Lobule 2 of animal CN276 2FD from brain slice 1. Similar to the one illustrated, this process of dividing the areas was carried out in all images to help reduce image noise and obtain a clearer view of the cells for the required image processing tasks. Another step developed before

the actual counting process was channel adjustment. As previously mentioned, namely, in subsection 2.4, microglia cells are associated with markers that help in their identification. The cell marker that helps in the identification of cells is the ionized calcium-binding adapter molecule 1, most commonly known as IBA1. As we can see in the image below, it is not easy to visualize the cells with the naked eye, even though it contains the IBA1 marker to help with its visualization. Knowing that these images are captured in a fluorescent format, channel separation was performed. The objective was to keep only the red channel for the quantification of cells, as this is where we can visualize them. Figure 26 illustrates the result of channel separation, and as we can see, it is now easier to identify the cells.

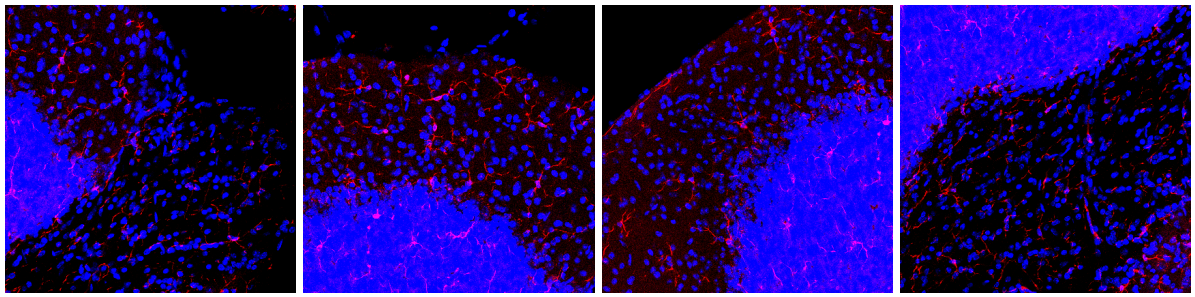


Figure 25: Microglial Cells - CN276 2FD - Slice 1 - Lobule 2 Images.

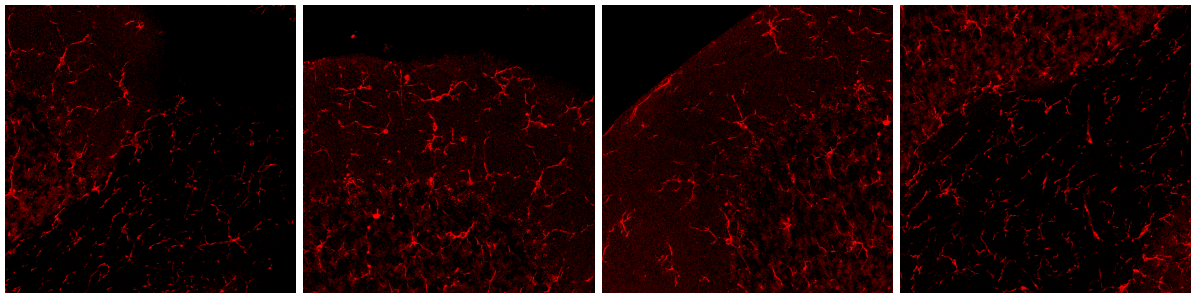


Figure 26: Microglial Cells - CN276 2FD - Slice 1 - Lobule 2 Red Channel Images.

Moving on to the quantification process itself, Table 19 displays the different settings applied to Lobule 2 of brain image 1 of animal CN276 2FD. It's important to mention that going forward, here is where the counting process varies from image to image. The purpose is to obtain the most accurate and precise results possible. Until now, the division of areas and the separation of channels have been done identically in all images. Threshold values vary in each experiment. A manual adjustment has been made with sliders to decrease the overlap as much as possible. As each image differs from the others, all areas of interest for the quantification are included with or without overlap. After properly adjusting the threshold value, as a good practice for the counting process came the application of the watershed algorithm. As it was possible to verify in the experiments conducted with the generic cell sample, the watershed algorithm separates with 1 pixel what it considers to be one or more overlapping cells, which is extremely important to obtain

exact results. All images were converted to binary and later to mask, to ensure that the algorithm can be properly applied. Finally, notice that in all the counts the circularity chosen for the objects ranges from 0.05 to 1.00, and the range of sizes is defined as 17 to infinity. It was possible to conclude that these values are the ideal ones through several trial and error tests that were validated with test counts. In this way, maintaining these values in all images, we ensure that we are not excluding any cell, as there are cells with different values of circularity and size in each image. To sum up, the protocol developed in this dissertation is composed of five steps, not counting the process of dividing the DCN areas and Lobules into multiple images. The phases that make up this protocol are the following: 1 - Channel Separation; 2 - Threshold Adjustment; 3 - Image Conversion; 4 - Cells Separation; 5 - Cells Quantification. The first stage is where we use the ImageJ functionality to separate channels, as the cells are only visible in the red channel. The second stage is where we guarantee all areas of interest are included with or without overlap, as we decreased the overlap as much as possible using the sliders for manual adjustment of the threshold values. Following this, the third stage is where the image is firstly converted to binary, resulting in an 8-bit image. Then the 8-bit image is converted to mask to ensure that the next phase can be successfully implemented. The fourth stage is the application of the watershed algorithm. Finally, to finalize the cell counting process comes the fifth phase of the developed protocol. Needs to be pointed out that if all the previous steps of the protocol are not followed, it is not possible to achieve good results. With the Analyze Particles functionality of ImageJ, we defined the previously discussed circularity and the size of objects, ensuring that we are not excluding any cell.

		1^{st}	2^{nd}	3^{rd}	4^{th}
Threshold	Min	0	0	0	0
	Max	1461	1750	1526	1574
		96.42%	96.26%	95.18%	95.04%
	Algorithm	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 19: CN276 2FD - Slice 1 - Analyze Particles - Quantification Settings for Lobule 2.

Focusing more on the second stage of the developed protocol, since every other phase is pretty much the same for every image, the 1^{st} experiment conducted in image 1 of Lobule 2 is the result of cell counting after applying manual thresholding to the image at a pixel value of 1461. Figure 27 illustrates the result of the counting process of this experiment. The 2^{nd} test presents the quantification settings of the conducted process in image 2 of Lobule 2 after once again applying manual thresholding to the image at a pixel value of 1750. The 3^{rd} experiment is related to the prior procedures. However, manual thresholding to image 3 of Lobule 2 was applied at a pixel value of 1526. The 4^{th} and final test for this Lobule is identical to the previous ones, as manual thresholding was again applied but at a pixel value of 1574. In the table, it is possible to verify which algorithm was used to threshold the image. In this case, the algorithm used was the Default. The reason for choosing this one was throughout the results obtained

in several trial-and-error quantification tests. It should be noted that when we adjust the threshold value in images, we are working with images in black and withe format. In total seven algorithms (Default, Huang, Intermodes, IsoData, MaxEntropy, Otsu and Yen) were tested. From those seven emerged the Default and the Intermodes algorithm. With better results and a little bit superior to the Intermodes algorithm, the Default algorithm proved to be ideal for the problem in question.

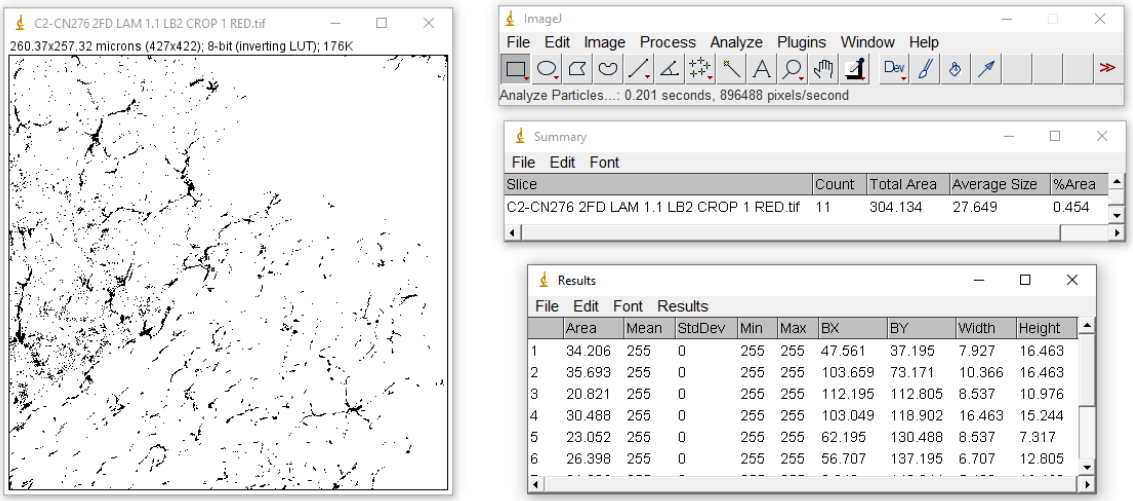


Figure 27: Microglial Cells - 1^{st} Image of Lobule 2 - Analyze Particles Automatic Quantification.

Tables 20, 21, 22 and 23 display the different quantification settings applied to Lobule 3, 4, 5 and 6, respectively. The remaining quantification settings for the remaining DCN areas and Lobules are displayed in Appendix A. The quantification settings of animal CN282 2TE, CN283 2FD and CN284 TDTE are presented in Appendix B, C and D, respectively. As can be verified, all the steps and values that led to obtaining the results regarding the automatic quantification of microglial cells process based on a more classical approach are fully documented. In this way, anyone who has access to these images and follows exactly all the steps can easily replicate all the results and consequently see their cell counting process optimized. How the entire protocol developed throughout this dissertation is documented, can serve as a basis for an application in a similar situation when dealing with a so-called more classical approach.

		1^{st}	2^{nd}	3^{rd}	4^{th}	5^{th}
Threshold	Min	0	0	0	0	0
	Max	1493	1911	1365	1477	1622
		96.63%	94.45%	95.96%	95.05%	95.36%
	Algorithm	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 20: CN276 2FD - Slice 1 - Analyze Particles - Quantification Settings for Lobule 3.

		1^{st}	2^{nd}	3^{rd}	4^{th}	5^{th}
Threshold	Min	0	0	0	0	0
	Max	2457	1847	1879	2104	2698
		96.85%	95.85%	96.71%	96.91%	96.63%
	Algorithm	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 21: CN276 2FD - Slice 1 - Analyze Particles - Quantification Settings for Lobule 4.

		1^{st}	2^{nd}	3^{rd}	4^{th}	5^{th}
Threshold	Min	0	0	0	0	0
	Max	2826	1895	2762	2875	2826
		97.52%	97.42%	97.25%	97.71%	97.33%
	Algorithm	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 22: CN276 2FD - Slice 1 - Analyze Particles - Quantification Settings for Lobule 5.

		1^{st}	2^{nd}	3^{rd}	4^{th}	5^{th}
Threshold	Min	0	0	0	0	0
	Max	2264	3276	2746	3276	2971
		97.57%	97.07%	96.02%	97.41%	98.39%
	Algorithm	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 23: CN276 2FD - Slice 1 - Analyze Particles - Quantification Settings for Lobule 6.

The results obtained and their respective discussion is presented in subsection 6.1.2. The approach considered optimal for the developed Analyze Particles protocol to automatically quantify microglial cells is sustained based on the results acquired with the quantification processes conducted.

ITCN

After the experiments carried out with the generic cell sample, detailed in subsection 5.2.1, the ITCN plugin was tested again to automate cell counting. Although the results obtained were not the best, due to the various factors presented and inherent to the counting process with this plugin, the experiments brought pertinent considerations to test this approach with microglial cells. The goal is to see if this approach is a valid alternative for the automatic quantification of cells based on a more classical methodology, and therefore a better alternative to manual counting processes. It is important to note that the images used were the same as those used in the developed Analyze Particles protocol.

As previously stated, this plugin requires a set of procedures to be applied to the image to make the count more reliable. Microglia cells are associated with markers that help their identification. The IBA1

marker in these cells is only visible in the red channel, so performing channel separation was needed. As the ITCN plugin doesn't deal very well with image noise, thresholding an image is also an important pre-procedure routine. Taking advantage of the threshold values applied in the Analyze Particles protocol, considered optimal given the results obtained in the counting process, these same values were reused for this counting process. To ensure that all cells are considered in the counting process, the watershed algorithm was applied. All images were converted to binary and later to mask, to ensure that the algorithm can work properly, and all areas of interest for the quantification are included. Moving on to the quantification process, Table 24 displays the different settings applied to Lobule 2 of brain image 1 of animal CN276 2FD within the ITCN plugin. The table also exhibits the values applied to threshold the image.

		1^{st}	2^{nd}	3^{rd}	4^{th}
Threshold	Min	0	0	0	0
	Max	1461	1750	1526	1574
		96.42%	96.26%	95.18%	95.04%
	Algorithm	Default	Default	Default	Default
ITCN	Cell Width	15	14	10	11
	Minimum Distance	19	39	55	47
	Threshold	3.0	2.5	5.5	6.5

Table 24: CN276 2FD - Slice 1 - ITCN - Quantification Settings for Lobule 2.

Focusing more on the ITCN part, as the other parts of this counting process were previously described, the 1^{st} experiment conducted in image 1 of Lobule 2 is the result of cell counting after defining the cell width in 15 pixels, the minimum distance between cells in 19 pixels and the ITCN threshold in 3.0. Figure 28 illustrates the counting process of this experiment. The 2^{nd} test presents the quantification settings of the conducted process in image 2 of Lobule 2 after specifying the cell width in 14 pixels, the minimum distance between cells in 39 pixels and the ITCN threshold in 2.5. The 3^{rd} experiment was conducted in image 3 of Lobule 2. The defined cell width is 10 pixels, the minimum distance between cells is 55 pixels, and the threshold value is 5.5. The 4^{th} and final test, performed in image 4 of Lobule 2, resulted in the definition of 11 pixels in the cell width, 47 pixels for the minimum distance between cells and 6.5 for the threshold value. It is important to point out that although the minimum distance between cells recommended is half the cell width, in the case of microglial cells, this recommendation was not followed. Microglial cells are distributed unevenly across the image and have different sizes. Following the recommendation would result in very poorly adjusted results and consequently would give rise to an imprecise counting process. Therefore, the minimum distance between cells was manually defined through a line that joins the two most close cells. This line is then measured to obtain the distance value in pixels. Similarly, the cell width is measured. A line joins the two most distant points of the cell nucleus and then this line is measured to obtain the cell width in pixels. The ITCN threshold value was also manually defined and differs from case to case to ensure that in each experiment the quantification is as accurate as possible. It should be noted that all the values presented were considered optimal for an adjusted counting process. To arrive at them

was necessary to carry out several experiments, more than twenty in the case of Lobule 2.

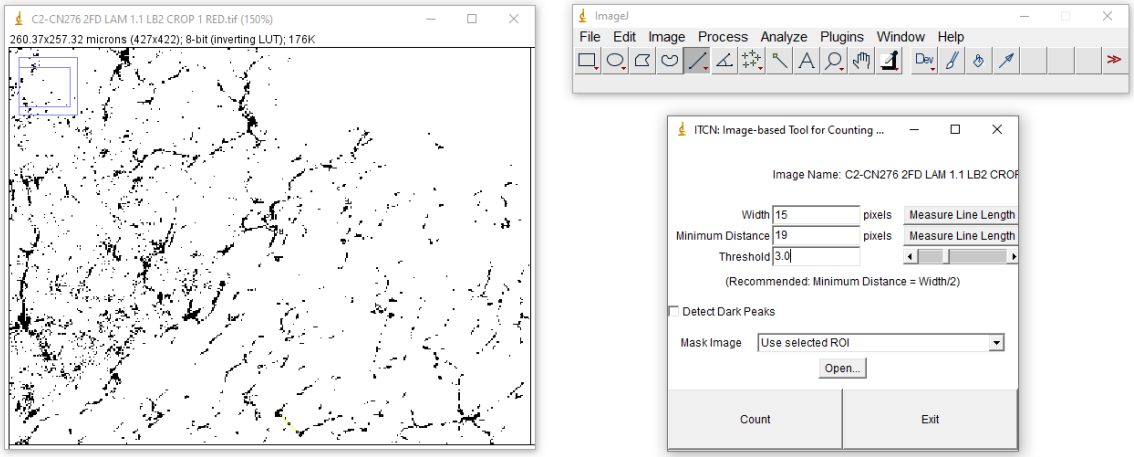


Figure 28: Microglial Cells - 1^{st} Image of Lobule 2 - ITCN Automatic Quantification.

Tables 25, 26, 27 and 28 display the different quantification settings applied to Lobule 3, 4, 5 and 6, respectively. Once again, anyone with these images can easily replicate all the results and see their cell counting procedure optimized. The way the entire process is documented serves as a basis for an application in a similar situation regarding a classical methodology.

		1^{st}	2^{nd}	3^{rd}	4^{th}	5^{th}
Threshold	Min	0	0	0	0	0
	Max	1493	1911	1365	1477	1622
		94.63%	94.45%	95.96%	95.05%	95.36%
	Algorithm	Default	Default	Default	Default	Default
ITCN	Cell Width	12	12	12	13	16
	Minimum Distance	27	21	24	43	55
	Threshold	4.5	5.0	5.5	4.0	1.5

Table 25: CN276 2FD - Slice 1 - ITCN - Quantification Settings for Lobule 3.

		1^{st}	2^{nd}	3^{rd}	4^{th}	5^{th}
Threshold	Min	0	0	0	0	0
	Max	1493	1911	1365	1477	1622
		94.63%	94.45%	95.96%	95.05%	95.36%
	Algorithm	Default	Default	Default	Default	Default
ITCN	Cell Width	16	15	20	22	22
	Minimum Distance	70	44	43	68	50
	Threshold	2.0	2.5	1.0	0.5	1.0

Table 26: CN276 2FD - Slice 1 - ITCN - Quantification Settings for Lobule 4.

		1^{st}	2^{nd}	3^{rd}	4^{th}	5^{th}
Threshold	Min	0	0	0	0	0
	Max	2826	1895	2762	2875	2826
		97.52%	97.42%	97.25%	97.71%	97.33%
	Algorithm	Default	Default	Default	Default	Default
ITCN	Cell Width	17	18	19	25	20
	Minimum Distance	50	80	77	60	50
	Threshold	1.5	0.5	0.5	0.5	1.0

Table 27: CN276 2FD - Slice 1 - ITCN - Quantification Settings for Lobule 5.

		1^{st}	2^{nd}	3^{rd}	4^{th}	5^{th}
Threshold	Min	0	0	0	0	0
	Max	2264	3276	2746	3276	2971
		97.57%	97.07%	96.02%	97.41%	98.39%
	Algorithm	Default	Default	Default	Default	Default
ITCN	Cell Width	20	16	23	22	13
	Minimum Distance	10	56	33	50	43
	Threshold	1.0	2.5	0.5	1.0	3.5

Table 28: CN276 2FD - Slice 1 - ITCN - Quantification Settings for Lobule 6.

The results obtained and their respective discussion is presented in subsection 6.1.2. This approach is discussed meticulously to sustain our opinion on the applicability of it to automatically quantify microglial cells based on the results acquired with the quantification processes conducted.

5.3 Deep Learning for Automatic Cell Counting

CNNs were chosen to develop a solution for automatic cell counting based on a deep-learning approach. As stated in subsection 3.4.3, CNNs are composed of a set of artificial neurons organized into layers such as the input layer, the hidden layer, and the output layer. They are also most commonly used to analyze visual imagery, as is the case in this dissertation work. In both approaches discussed in subsection 3.4.6, the common point between both is the use of CNNs. In this way, and after having also analyzed all relevant documentation and literature, the use of CNNs to solve the problem of automated microglial cell counts based on a deep learning approach seems to be ideal. The programming language used in this approach was Python, which is a high-level programming language. The version used was 3.7.15. Different libraries were also used in the design of this solution, such as the open source library for ML TensorFlow (version 2.9.2), the high-level NN library that runs on top of TensorFlow as is the case of Keras (version 2.9.0), the software library for creating graphs and general data visualizations Matplotlib (version 3.2.2), the software library created for the Python language for data manipulation and analysis Pandas (version 1.3.5), the Python library that supports processing of large, multi-dimensional arrays and matrices, along with a large collection of mathematical functions NumPy (version 1.21.6) and the open

source machine learning library for the Python programming language Scikit-learn (version 0.24.1). In order not to affect and influence the results obtained in both approaches studied, within the deep learning methodology, the version of the programing language and libraries was the same in both cases.

5.3.1 Cell Colony Sample

Once again, to fully comprehend the best approach to automatically count cells within a deep learning-based methodology the generic cell sample was selected to test two approaches. The goal was to understand the extensiveness of these two strategies, namely the pros and cons of applying a CNNs to a classification approach. In this case, the models with CNNs were applied to a cell classification procedure based on the number of cells present in the image and a classification technique based on the area occupied by the cells in the sample. It is important to point out that according to what was highlighted in section 4.1, for a model to work properly and have adjusted results, the generic cell image was divided into 16 equal images to compose a dataset with 16 samples.

Classification per Number of Cells

The objective of a classification approach is to classify one or several images using previously defined classes. Table 29 presents the classes defined for the classification of the generic cell sample based on the number of cells. To correctly define the intervals, in all 16 images the cells present were counted using the Analyze Particles protocol developed in this dissertation. Thus, an image that does not contain more than 40 cells will be classified as "Few", given the low number of cells. An image that contains more than 40 cells and less than 50 is classified as "Average". Finally, an image that has more than 50 cells will be classified as "Many", given the high number of cells.

Label	Interval
Few	< 40
Average	$\geqslant 40$ & < 50
Many	$\geqslant 50$

Table 29: Cell Colony Sample - Labelling Classes (Number).

Once the intervals for cell classification were defined and the cells were counted, labelling the images is required. Figure 29 displays the labelling performed on the images. In total 3 images were labelled as "Few", 8 as "Average", and 5 as "Many". At the level of the code structure for the construction of the developed CNNs model, after labelling the images it was necessary to do the load data, load labels and define the classes. Following the TensorFlow documentation, the appropriate CNNs model was created to help solve the problem presented in this dissertation. Once the model was created, it was necessary to compile and fit it. For that, in this compile and fit function we passed the model type, the training and test set, the batch size, the number of epochs, the value of the learning rate, the application of data augmentation and the plot of the learning curves initialized as false. The metric defined for the model

is "accuracy". Within this function, the number of splits was defined. In this specific case, the number of splits defined was 16, so we can cross-validate with 16 folds and the error will be the average of 16 folds. As we have little data to train and test, we try to maximize the training data since in each fold we train with 15 images and test only with 1 image. This was done for the 16 images of the dataset, which makes training and testing more time-consuming, but will generate better results. Then it was necessary to plot the learning curves. The data was prepared again by reading the training and test set and the classes. To plot the learning curves and understand how the model behaves, the most complex values were assigned to the parameters. The values in question were one for the batch size, one hundred for the number of epochs, and 0.0005 for the learning rate. Data augmentation was used in this case. The number of folds for the plot of learning curves was four, meaning that we train with twelve cases to predict four, thus bringing even more complexity to the model. It is important to point out that to guarantee that these results can be replicated on any machine by any user, a seed has been defined. Thus, similar to the classical methodology, the processes developed in an approach based on deep learning can also be fully replicated.

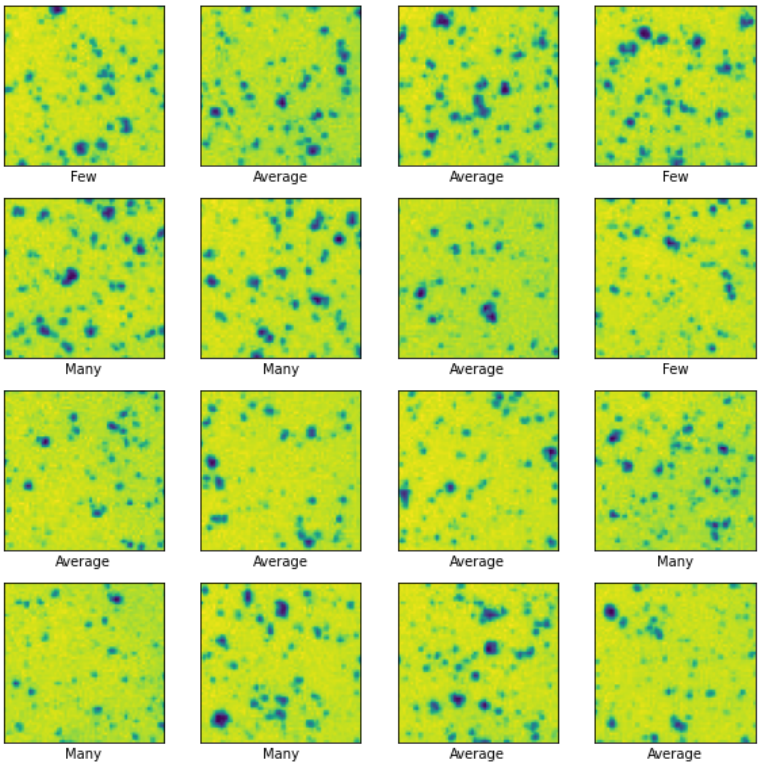


Figure 29: Cell Colony Sample - Image Labelling (Number).

The purpose of learning curves is to help us understand the optimal values for the model, namely the number of epochs. Figure 30 illustrates the learning curves obtained in this approach. We can conclude by reading the plots that the ideal value for the number of epochs goes up to 35. Going beyond this value is irrelevant to the model since the training loss and validation loss are getting further and further away

from each other. Since our dataset only consists of 16 images, going beyond 16 in the value assigned to the batch size would also be irrelevant. So, to be able to tune the model, the following list of values was defined for the different parameters:

- batch_size_list = [1, 2, 4, 8, 16];
- epochs_list = [1, 2, 3, 4, 5, 10, 15, 20, 25, 30, 35];
- learning_rate_list = [0.0005, 0.005, 0.001];
- apply_data_augmentation_list = [True, False].

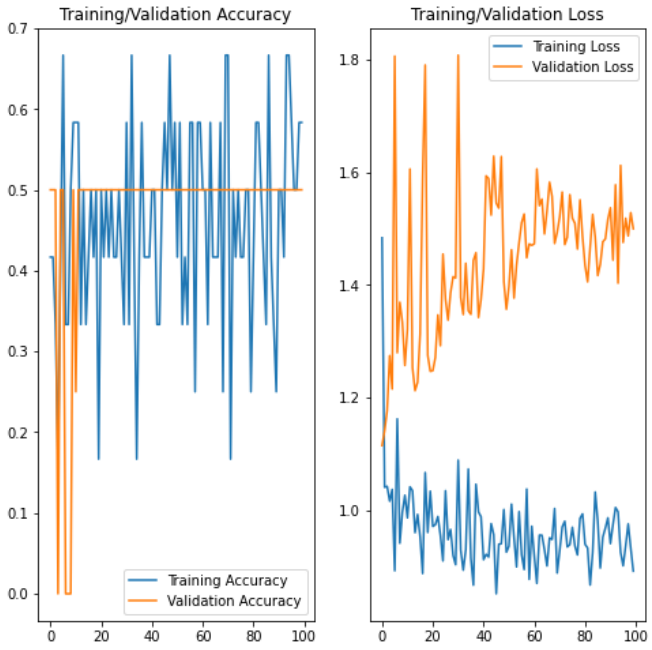


Figure 30: Cell Colony Sample - Learning Curve (Number).

Several experiments were carried out, with the different values defined in the parameters, totalling 330. The results obtained and a respective discussion is presented in subsection 6.2.1. The approaches considered optimal to help in the problem of the microglial cells are explained.

Classification per Area of Cells

Similar to the classification approach per number of cells, the objective is again to classify one or several images using previously defined classes. Table 30 presents the classes defined for the classification of the generic cell sample based on the area of cells. To properly define the intervals, in all 16 samples the cells were counted using the Analyze Particles protocol developed in this dissertation. By using this protocol and through the counting results, we can access the percentage of the area that these cells occupy in the image. Thus, an image that does not contain a percentage of cell area greater than 5.2 will be classified

as "Few". An image that contains a percentage of cell area greater than 5.2 and less than 6.9 is classified as "Average". Finally, an image that contains a percentage of cell area greater than 6.9 will be classified as "Many".

Label	Interval
Few	< 5.2
Average	$\geqslant 5.2$ & < 6.9
Many	$\geqslant 6.9$

Table 30: Cell Colony Sample - Labelling Classes (Area).

Once the intervals for cell classification were defined and the area of the cells was measured, the first step for classification was the labelling of the images. Figure 31 displays the labelling performed on the images. In total 4 images were labelled as "Few", 6 as "Average", and 6 as "Many". Then, it was necessary to do the load data, load labels and define the classes in the model. Leveraging the previous procedure, it was necessary to compile and fit it, passing the model type, the training and test set, and the respective parameters. As the objective is classification, the chosen metric was "accuracy". The number of splits defined was the same as in the previous experience. With this number of splits, we can cross-validate with 16 folds. The error will be an average of 16 folds. For the plot of learning curves, the same procedure as the previous approach was followed.

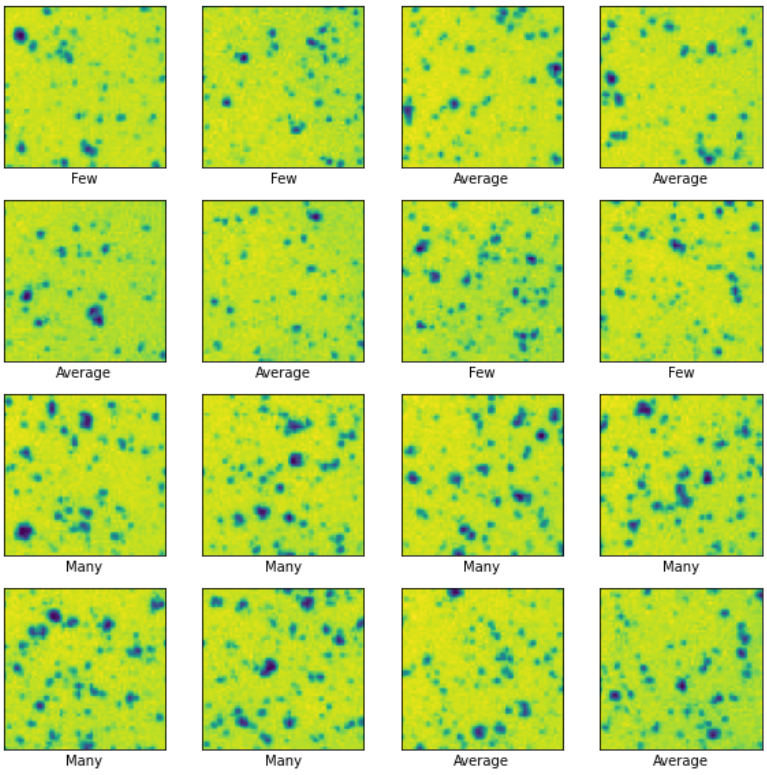


Figure 31: Cell Colony Sample - Image Labelling (Area).

Before moving on to the analysis of learning curves, it is necessary to point out that once again to guarantee that these results can be replicated on any machine by any user, a seed has been defined in the code. The purpose of learning curves, in this case, is the same as in the previous approach, which is to help us understand the optimal number of epochs for the model. Figure 32 illustrates the learning curves obtained in this approach. Through their analysis, we can see that there is an approximation between the line of training loss and validation loss up to 20 epochs, and then they distance themselves from each other. The point of going up to a relatively high number of epochs with the learning curves plot is to see if both lines of the loss come together again at some point, which was the case when the number of epochs is 60. Going beyond 95 epochs is irrelevant to the model since the training loss and validation loss start getting further and further away from each other. Once again, knowing that our dataset only consists of 16 images, going beyond 16 in the value assigned to the batch size would also be irrelevant. To be able to tune the model, the following list of values for the model parameters was defined:

- batch_size_list = [1, 2, 4, 8, 16];
- epochs_list = [1, 2, 3, 4, 5, 10, 15, 20, 60, 65, 70, 75, 80, 85, 90, 95];
- learning_rate_list = [0.0005, 0.005, 0.001];
- apply_data_augmentation_list = [True, False].

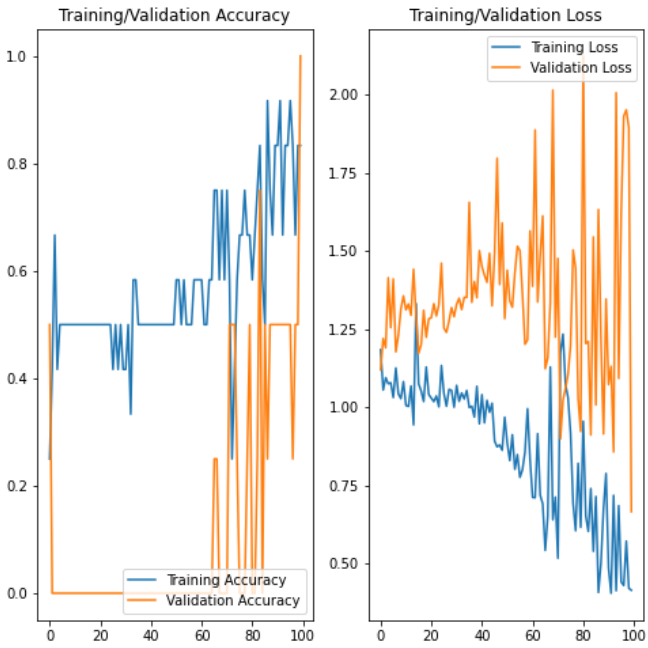


Figure 32: Cell Colony Sample - Learning Curve (Area).

In total, 480 experiments were carried out, with the different values defined in the parameters. The results obtained and a respective discussion is presented in subsection 6.2.1. The best approaches are

described based on the results acquired from the experiments conducted with the generic cell sample are explained based on the results acquired.

5.3.2 Brain Image Samples

The objective of this dissertation is to automate the cell counting process, namely the quantification of microglial cells. As already explained, these cells are associated with the markers that help in their identification. An approach to automate the microglia cell counting process is image classification. According to relevant literature and documentation, namely, the approaches presented in subsection 3.3.3, the use of a model based on CNNs proves to be an ideal solution for the problem. The experiences carried out with the cell colony detailed in subsection 5.3.1 also came to attest that the conclusions drawn from the articles brought for discussion were correct, namely that the use of CNNs is ideal for automating the microglial cell counting process. Both the approaches presented and those developed fetched good practices that need to be considered in our case. Therefore, using a model based on CNNs two approaches that aim to automate cell counting were applied to the brain images. The goal is to understand the extensiveness of these two strategies, namely their respective pros and cons. Taking advantage of the work produced within the classical approach, the division performed on the images was reused, thus composing a dataset with 661 images. With this number of images, we guarantee that the model can work properly and have adjusted results. Next, the experiments conducted with these images, namely the classification approach based on the number of cells in an image and the percentage of area that the cells occupy in an image are explained.

Classification per Number of Cells

Once again the objective of a classification approach is to classify one or several images using previously defined classes. Analyzing the entire counting process and the results inherent to the experiments carried out with the generic cell sample, the classification approach based on the number of cells has proven to be a very valid alternative for automating the quantification of microglial cells. That said, Table 31 presents the classes defined. To correctly define the intervals, we resorted to the cell count results for each image, obtained from the Analyze Particles protocol developed in this dissertation. Thus, an image that does not contain more than 15 cells will be classified as "Few". An image that contains more than 15 cells and less than 25 is classified as "Average". Finally, an image that has more than 25 cells will be classified as "Many".

Label	Interval
Few	< 15
Average	$\geqslant 15$ & < 25
Many	$\geqslant 25$

Table 31: Microglial Cells - Labelling Classes (Number).

Having the intervals for cell classification defined and the cells counted, image labelling is the next step required. Figure 33 displays part of the labelling performed on the images. In total 339 were labelled as "Few", 203 as "Average", and 119 as "Many". In terms of the code structure for building the model, the approach followed was very similar to the one implemented with the cell colony. Once again, following the TensorFlow documentation, the appropriate CNNs model was developed to help automate microglial cell counting. The model was created so it was necessary to compile and fit it. For that, and since excellent results were obtained previously, the same parameters were passed to the model. The parameters in question were the following: the model type, the training and test set, the batch size, the number of epochs, the value of the learning rate, and the application of data augmentation. The data augmentation value has been initialized to false. As in the previous cases, the metric defined for the model is "accuracy" since it is a classification approach. The number of splits for the model was also defined, which in this specific case was 5. We can do cross-validation with 5 folds. The error will be the average of the 5 folds. In this way, we don't add too much difficulty to the model and manage to have an equally effective but faster process. That said, we are left with 529 images for training and one 132 for testing. This is done for each fold, which makes training and testing a little more time-consuming, but will generate more reasonable results.

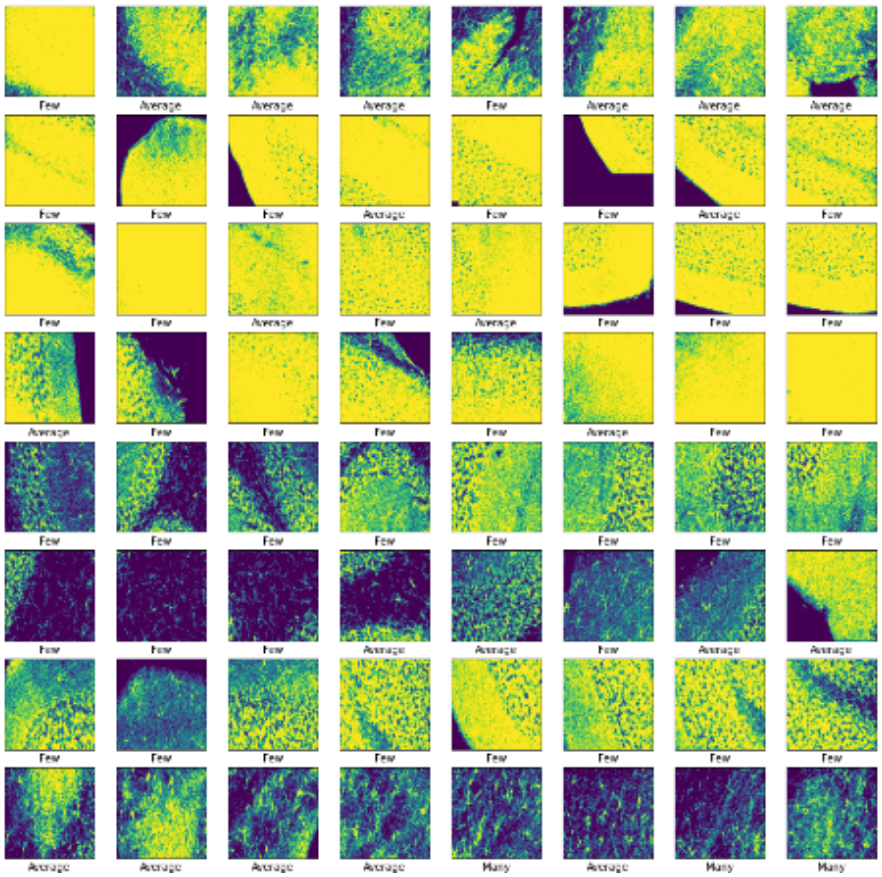


Figure 33: Microglial Cells - Image Labelling (Number).

After creating and defining the model was necessary to plot the learning curves. As expected, the data was prepared by reading the training and test set and the classes. To understand how the model behaves, the most complex values were assigned to the parameters to plot the learning curves. The values in question were one for the batch size, 50 for the number of epochs, 0.0005 for the learning rate, and data augmentation has been initialized to true. The number of folds for the plot of learning curves was 4, meaning that we train with 496 cases to predict one 165. This adds complexity to the model and gives us an idea of how it will behave. Once again, it is important to point out that to guarantee that these results can be replicated on any machine by any user, a seed has been defined.

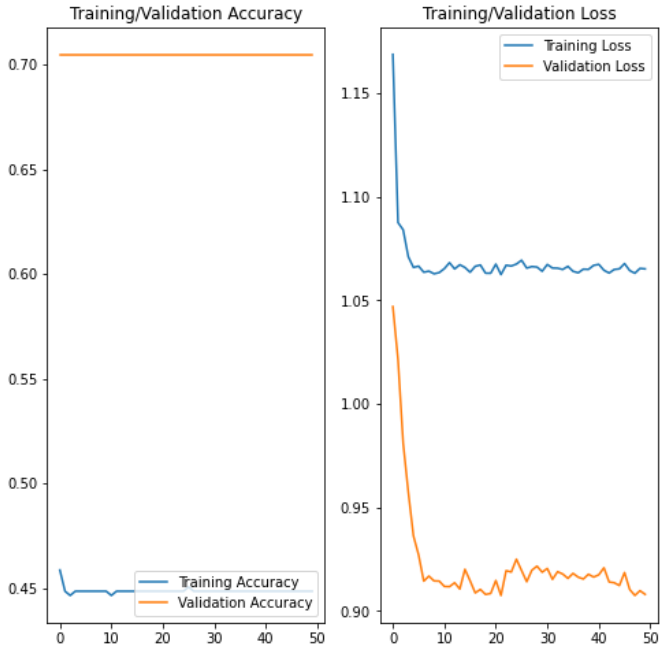


Figure 34: Microglial Cells - Learning Curve (Number).

Moving on to a more detailed analysis of the results obtained with the implementation of learning curves, we came to the idea that the ideal value for the number of epochs goes up to 20. Figure 34 illustrates the learning curves obtained in this approach. As explained, the purpose of learning curves is to help us understand the optimal values for the model, namely the number of epochs. As the lines of training loss and validation loss are getting further and further away from each other when the number of epochs is higher than 20 is irrelevant to go above that value. Knowing that our dataset is composed of 661, in this case, it is already relevant to increase the maximum value for the batch size to 32. So, to tune the model, the following list of values was defined for the different parameters:

- batch_size_list = [1, 2, 4, 8, 16, 32];
- epochs_list = [1, 2, 3, 4, 5, 10, 15, 20];
- learning_rate_list = [0.0005, 0.005, 0.001];

- apply_data_augmentation_list = [True, False].

In total, 288 experiments were carried out, with the different values defined in the parameters. The results obtained and a respective discussion is presented in subsection 6.2.2. The best approaches are described based on the results acquired from the experiments conducted. The procedure considered optimal that helps to automate the microglial cell counting is described in detail as it will be brought for discussion within the various methodologies, manual, classic and deep learning.

Classification per Area of Cells

Another approach that helps automate the cell counting process is the classification technique based on the area that cells occupy in an image. Similar to the previous procedure, the objective is again to classify one or several samples using previously defined classes. Table 30 presents the defined classes. To correctly define the intervals, we resorted again to the cell count results obtained from the Analyze Particles protocol developed in this dissertation. Thus, an image that does not contain a percentage of cell area greater than 0.45 will be classified as "Few". An image that contains a percentage of cell area greater than 0.45 and less than 0.85 is classified as "Average". Finally, an image that contains a percentage of cell area greater than 6.9 will be classified as "Many".

Label	Interval
Few	< 0.45
Average	$\geqslant 0.45$ & < 0.85
Many	$\geqslant 0.85$

Table 32: Microglial Cells - Labelling Classes (Area).

Once defined the intervals for cell classification and the area of the cells was measured, the second procedure was image labelling. Figure 33 displays part of the labelling performed on the dataset. In total 324 images were labelled as "Few", 270 as "Average", and 67 as "Many". Then, following the same approach as the previous procedure, it was necessary to do the load data, load labels and define the classes in the model. Up next, was necessary to compile and fit it CNNs model, passing the model type, the training and test set, and the respective parameters. Leveraging the previous procedure, the number of splits defined was the same as in the previous experience since excellent results were obtained. With 5 splits we can do cross-validation with 5 folds, and the error will be the average of the 5 folds.

To understand how the model behaves was necessary to plot the learning curves. Therefore, the most complex values were assigned to the different parameters. The values in question were the same as in the previous procedure and were: one for the batch size, 50 for the number of epochs, 0.0005 for the learning rate, and data augmentation has been initialized to true. The number of folds for the learning curves plot was also the same since in this way we added complexity as much as possible to the model and gives us an idea of how it will behave. A seed was again created to ensure that these results can be replicated on any machine by any user.

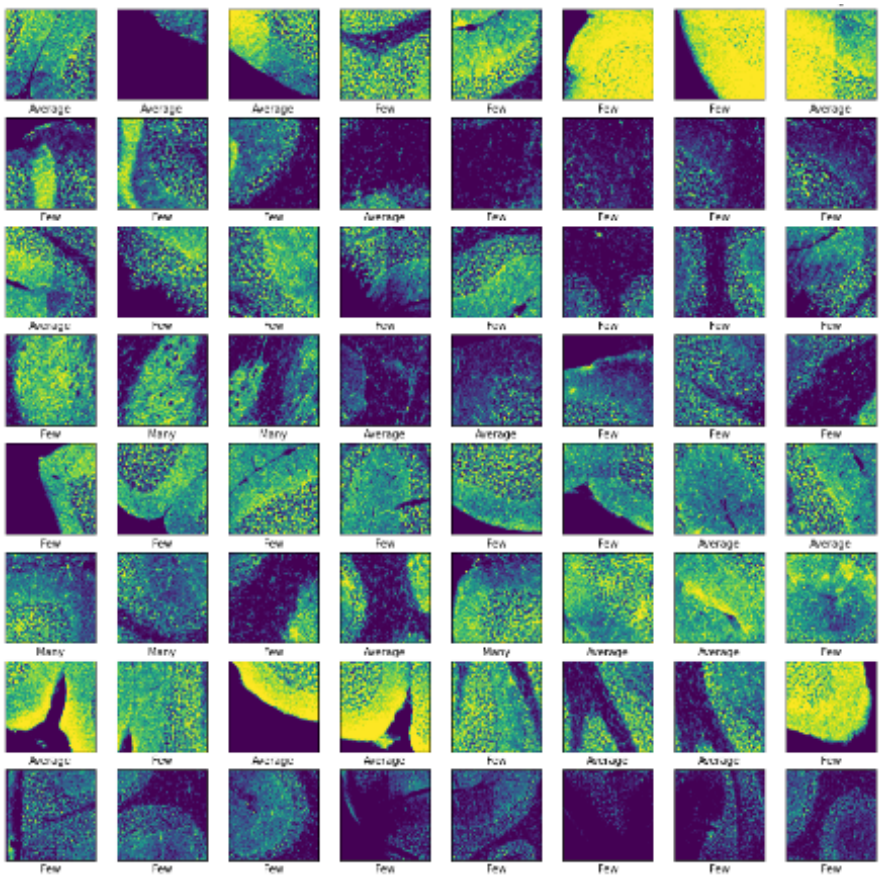


Figure 35: Microglial Cells - Image Labelling (Area).

Analyzing the plots we concluded that the ideal value for the number of epochs goes up to 25. Figure 34 illustrates the learning curves obtained in this procedure. Once again, the purpose of learning curves is to help us understand the optimal values for the model, namely the number of epochs. After 25, the lines of training and validation loss are getting further and further away from each other so it is not relevant to go beyond the 25 epochs. As with the previous approach, the maximum value for the batch size to 32 since the dataset is composed of 661 samples. To tune the model, the following list of values was defined for the different parameters:

- batch_size_list = [1, 2, 4, 8, 16, 32];
- epochs_list = [1, 2, 3, 4, 5, 10, 15, 20, 25];
- learning_rate_list = [0.0005, 0.005, 0.001];
- apply_data_augmentation_list = [True, False].

In total, 324 experiments were carried out. The results obtained and a respective discussion is presented in subsection 6.2.2. Once again, the best experiences are described based on the results acquired from the experiments conducted. The best approach is described in detail since it can be a solution to

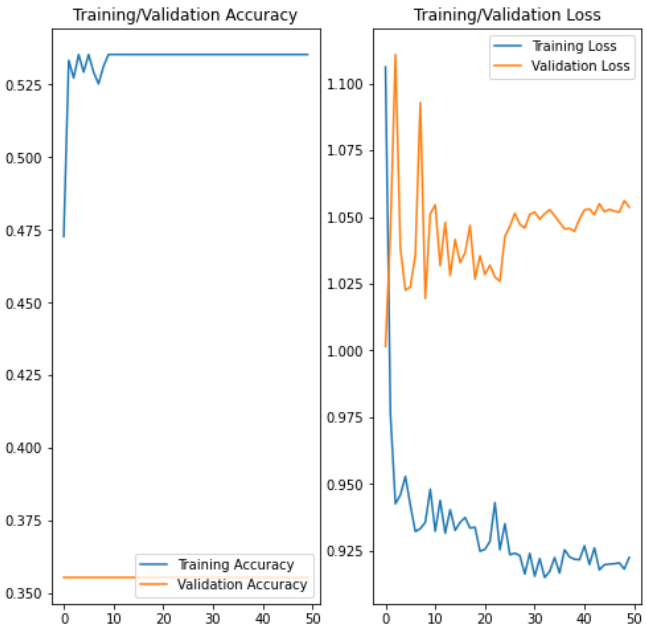


Figure 36: Microglial Cells - Learning Curve (Area).

automate microglial cell counting. This procedure will also be brought for discussion within the various methodologies, manual, classic and deep learning.

6

Results and Discussion

This chapter presents the reader with all the results of the automated counting experiences performed with the different approaches. Initially, the results of the counting process of the generic cell sample are presented followed by the results of the quantification of microglial cells. Finally, based on the results gathered from the overall experience between different approaches and methodologies, a discussion is presented where all the pros and cons of each procedure are raised since the goal is to find the most suitable solution to the microglial cell automatic quantification problem.

6.1 Classic Methods for Automatic Cell Counting

Based on ImageJ, which was the software of choice for automatic cell counting, in terms of a more classic methodology, several experiments were carried out, as detailed in the previous chapter. The experiments conducted with a generic cell sample aimed to understand the possible limitations of the software, as well as to study the selected approaches, namely the Analyze Particles functionality and the ITCN Plugin. So, through the results obtained, a discussion is presented in subsection 6.1.1. This discussion made it possible to conclude what should or should not be done with each one of these approaches in terms of microglial cell counts. Subsection 6.1.2 details the results obtained from the automatic quantification of microglial cells, as well as, presents the discussion of them, and bases the procedure considered optimal, chosen for the comparison between the different approaches, the manual, the classic and the deep learning approach.

6.1.1 Cell Colony Sample

As was already explained, this sample was selected to understand the limitations of ImageJ. According to all relevant literature and documentation studied in the state-of-the-art, two different approaches were tested within the classical methodology. Next, the quantification results gathered from the counts performed are presented. For each strategy, results are discussed, followed by the conclusions drawn that will positively influence the microglial cell counting process.

Analyze Particles

To study the best approach to automatically quantify the number of cells in an image, with the Analyze Particles functionality, 5 test experiments were conducted. Table 33 presents the results obtained in the counting process. According to the information presented in subsection 5.2.1, in all experiments, care was taken to test different approaches within the developed protocol, except for the values assigned to the size and circularity of the cells. Behind this exception is the fact that the cells present in the sample have different sizes and circularity from each other, so changing and defining a specific size would influence the results and leave some cells out of the count. At first glance, we may think that we will have a lot of noise in the image and that any pixel will be counted as a cell but we couldn't be more wrong. Throughout the developed protocol, and the steps implemented before the counting process itself, such as removing the image background, and adjusting the threshold value, among others, we are reducing noise and eliminating image outliers. In this way, we obtain a more accurate and reliable counting process. It is easy to notice that the algorithm used to change the threshold value in the image was always the same, and the reason for its choice was, according to the analysed documentation, because it is the best one to deal with image analysis in the black and white format.

Test Experience	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	710	10979	15.463	6.628
2	690	10126	14.675	6.113
3	710	10079	14.196	6.085
4	881	8120	9.217	4.902
5	893	8094	9.064	4.886

Table 33: Cell Colony Sample - Analyze Particles - Automatic Quantification Results.

Moving on to the results of the conducted experiences, the counting process of the 1^{st} experiment resulted in 710 cells quantified. It should be noted that in this experiment, before the counting process itself, only the threshold of the image was treated. Therefore, it was possible to understand how the counting process behaves without removing the background or separating cells. The count performed in the 2^{nd} experiment quantified 690 cells. In this case, before the threshold treatment, the background of the image was removed, as explained in the previous chapter. Thus, and different from the previous test, it was possible to see how the count behaves when we just remove the background and treat the threshold. The 3^{rd} experience resulted in the quantification of 710 cells. In addition to background removal and threshold processing, in this experiment, the image was converted to a binary format, namely an 8-bit format, to be able to apply the watershed algorithm. As explained, this algorithm separates with 1 pixel what it considers to be two cells or more together. Thus, instead of quantifying just one cell when there may be two or more, it quantifies the correct number of cells, two or more. The 4^{th} test resulted in the quantification of 881 cells. Similar to the counting process in the 2^{nd} experiment, this one aimed to understand what happens when we change the radius of the "rolling ball"in the background removal

process. The last test resulted in 893 cells quantified. Also with a low value of the "rolling ball"radius for background removal, in this experiment, the image was converted again to a binary format to be able to apply the watershed algorithm. In this way, with this and with the previous experience, we were able to understand how the count behaves with low values in the background removal and consequently greater noise. To try to minimize the outliers, in the 5^{th} test, the separation of cells that the software judges to be together was tested as was explained. Auto-thresholding was applied both in this experiment and in the previous one.

To conclude, of all the experiments carried out the first three should be highlighted. The counting results are very approximate and accurate. In the case of experiments 1^{st} and 3^{rd}, we even reached the same result in the cell counting process. Through the results of these two experiments, we learned more pros than cons regarding the Analyze Particles functionality. Thresholding an image proved to be a very important task as it helps in removing noise and makes the counting process more accurate. In the case of this colony cell sample, the image is not very complex, hence auto-thresholding has been used. In the case of images of microglial cells, we have more complex images due to the complexity of the cell itself, so it can be concluded that auto-thresholding should not be the ideal solution. So, it will be necessary to adjust the threshold manually through the sliders decreasing the overlap as much as possible. From these experiments, it was also possible to conclude that removing the background from the image can be beneficial for the counting process. When we compared the background removal process through the passing of a "rolling ball"performed in experiments 2^{nd} and 3^{rd} with experiments 4^{th} and 5^{th}, we noticed that with a higher radius value the counting process is more accurate. Finally, using the algorithm that separates what is considered to be cells together is something very positive that this protocol can bring to the quantification of microglial cells, as it was possible to prove with the results of this sample. The only necessary care is to transform the image to a binary format, otherwise, it is not possible to apply this algorithm. That said, the Analyze Particles functionality brings many benefits to the automatic quantification of cells, as it makes a process that is usually time-consuming simpler, faster and more effective. Within the classical methodologies, it is a reliable solution to the problem presented and therefore it will be used in the counting of microglial cells.

ITCN Runner

Another approach within the classic methodology to automatically quantify the number of cells in an image is the ITCN plugin. With the ITCN plugin, 5 experiments were performed. Table 34 presents the results obtained throughout the counting process. As mentioned in subsection 5.2.1, in all experiments, different approaches within the ITCN plugin were tested to verify the effectiveness of this approach. This protocol requires certain procedures applied to the image before the application of the ITCN. Taking advantage of the Analyze Particles experiments, some practices were reused. As in the previous approach, all images were converted to binary and later to mask, to ensure that all areas of interest for the quantification are included. Throughout this pre-procedure task, such as removing the image background and adjusting the threshold value, we are reducing noise and eliminate image outliers. In this way, we obtain a more

accurate and reliable counting process.

Test Experience	Number of Cells
1	286
2	1136
3	908
4	981
5	969

Table 34: Cell Colony Sample - ITCN - Automatic Quantification Results.

Moving on to the results of the conducted experiences, the counting process of the 1^{st} experiment resulted in 286 cells quantified. The result is far from the actual number of cells in the image. However, should be noted that in this experiment, before the quantification, only the background was subtracted by applying auto-thresholding. Another fact that may help to explain this result is the defined size for the cells, which may not have been ideal since cells with a size smaller than 9 pixels are not counted. The minimum distance between cells was 4 pixels contrary to what is recommended by the plugin. The plugin always recommends that the minimum distance between cells is half the size of the smallest cell. As the cells are dispersed non-uniformly across the image, following the recommendation we would have many outliers in the counting process. The count performed in the 2^{nd} test is similar to the previous one, as the only difference was the ITCN threshold value. The objective was to understand how the counting process behaves when we adjust the threshold value. This resulted in 1136 cells quantified. The 3^{rd} experience is the most accurate of all and resulted in the quantification of 908 cells. Once again, similar to the previous ones concerning the pre-counting procedures, in this experiment, the explanation for the results obtained comes from the definition of a smaller size for the cells and the value of the ITCN threshold was increased. To test the benefits of pre-counting procedures, in the 4^{th} test the background was subtracted after the passage of a "rolling ball". The image was then converted to binary for the application of the watershed algorithm. The size defined for the cells was again reduced and consequently, it was necessary to increase the threshold value. This resulted in 981 cells being counted. In the 5^{th} experience the size defined for the cells was maintained, and the minimum distance between cells was changed to a value smaller by 1 pixel. The threshold value was increased again, having counted 969 cells.

To conclude, of all the experiments carried out the 3^{rd} is the one that should be highlighted since was the one that had the closest results to the actual number of cells. Through the results of these experiments, we learned more about the quantification process with the ITCN plugin. Pre-quantification procedures are extremely important, and the more rigorous they are, the more effective the counting process will be. Another very important aspect to consider with this approach is the definition of cell size and the minimum distance between cells. If these two parameters are not correctly defined, the counting process is not efficient because it will either count more or fewer cells. The value assigned to the ITCN threshold is also important as it helps to deal with image noise and outliers. Thus, to achieve interesting results

that are close to reality with this approach, it is necessary to carry out several experiments, contrary to the approach taken with the Analyze Particles. Despite this, and sometimes taking longer when compared to the previous procedure, this approach undoubtedly manages to automate the cell counting process. That said, the ITCN plugin brings some benefits to the automatic quantification of cells. Within the classical methodologies, it can be a reliable solution to the problem presented and therefore it will be tested in the quantification of microglial cells.

6.1.2 Brain Image Samples

As explained, to obtain more accurate results, and as microglial cells and the marker associated with them are only visible in the red channel, all DCN areas and Lobules were divided into several images with the same size, to minimize image noise and obtain a clearer view of the cells. Through the literary review and the conclusions drawn from previous approaches with cell colony sample, ImageJ offers the possibility of effectively counting objects in 2D images. In this way, ImageJ fits perfectly into the context of this dissertation, where two alternatives emerge to automate the microglial cell counting process. Next, for each strategy, results are presented and discussed, followed by the conclusions drawn that will positively influence an optimal approach for the automation of the microglial cell counting process.

Analyze Particles

According to relevant literature and documentation and also through the results obtained with the generic cell sample-related experiments, an approach to automate the microglial cell quantification based on a more classical approach is the developed Analyze Particles protocol. Taking advantage of the experiences that obtained excellent results with the cell colony sample, and the good practices of articles related to similar problems, brought up for discussion the protocol presented in subsection 5.2.2 was developed/created. Table 35 presents the results obtained in the counting process of Lobule 2 of animal CN276 2FD. In all quantification experiments, care was taken to implement the best approach within the developed protocol. For this, and as already explained, this protocol consists of a set of steps that, when respected, allow the automation of the cell counting process. This makes the whole process faster and the performance obtained in some cases turns out to be superior to manual approaches.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	11	304.134	27.649	0.454
2	19	443.560	23.345	0.662
3	19	503.421	26.496	0.751
4	16	369.572	23.098	0.552

Table 35: CN276 2FD - Slice 1 - Analyze Particles - Automatic Quantification Results for Lobule 2.

Moving on to the results of the quantification process itself, in particular, those conducted in Lobule 2

of animal CN276 2FD, since re-emphasizing the pre-quantitation procedures would bring repetitive information as they are properly documented in subsection 5.2.2. The 1^{st} image quantification process resulted in 11 quantified microglial cells. It should be noted that in this experiment, the image was thresholded at a pixel value of 1461. The 2^{nd} cell counting process resulted in 19 cells quantified, after thresholding the image at a pixel value of 1750. The 3^{rd} resulted once again in 19 cells quantified, and the 4^{th} resulted in 16 microglial cells quantified. As expected, both images were thresholded at a pixel value of 1526 and 1574, respectively. Analyzing together the results obtained in the four images, since they make up the Lobule 2 of the CN276 2FD animal, the performance obtained is on par with the manual approach since the number of cells counted in both cases was similar. It should be noted that even with a very similar performance, this process was much faster and more accurate since the error associated with the user is minimal. Tables 36, 37, 38 and 39 display the different results of the quantification process of Lobule 3, 4, 5 and 6, respectively. The remaining quantification settings for the remaining DCN areas and Lobules are displayed in Appendix E. The quantification settings of animal CN282 2TE, CN283 2FD and CN284 TDTE are presented in Appendix F, G and H, respectively. As can be verified, all results regarding the automatic quantification of microglial cells process based on a more classical approach are fully documented. In this way, anyone who has access to these images and follows exactly all the steps can easily replicate all the results and consequently see their cell counting process optimized. The documentation of this dissertation can serve as a basis for an application in a similar situation when dealing with a so-called more classical approach.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	21	528.703	25.176	0.789
2	18	423.483	23.527	0.632
3	10	209.325	20.932	0.312
4	16	405.637	25.352	0.625
5	10	276.993	27.699	0.413

Table 36: CN276 2FD - Slice 1 - Analyze Particles - Automatic Quantification Results for Lobule 3.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	6	169.542	28.257	0.253
2	16	400.803	25.050	0.598
3	14	299.673	21.405	0.447
4	9	212.299	23.589	0.317
5	10	245.761	24.576	0.367

Table 37: CN276 2FD - Slice 1 - Analyze Particles - Automatic Quantification Results for Lobule 4.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	12	269.557	22.463	0.402
2	9	195.940	21.771	0.292
3	14	303.019	21.644	0.452
4	4	88.489	22.122	0.132
5	9	202.632	22.515	0.302

Table 38: CN276 2FD - Slice 1 - Analyze Particles - Automatic Quantification Results for Lobule 5.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	7	157.644	22.521	0.235
2	6	130.131	21.688	0.194
3	20	444.676	22.234	0.664
4	7	149.465	21.352	0.223
5	5	122.323	24.465	0.183

Table 39: CN276 2FD - Slice 1 - Analyze Particles - Automatic Quantification Results for Lobule 6.

To conclude, the results obtained in the quantification of microglial cells proved that the protocol developed is a viable alternative to manual approaches. To support an optimal approach to the problem presented in this dissertation, we highlight not only one experience but the entire protocol. We know that brain images of microglial cells are quite complex which in itself does not help in the counting process. It could be expected that the results obtained would not be comparable to manual processes, as they dictate the most common approaches within the clinical context. This protocol, and the results related to it, came to contradict this premise. When image processing and analysis tasks are performed correctly, accurately and appropriately, the results speak for themselves. As already mentioned, the results obtained are comparable to those of manual approaches, even in some cases we believe that they were more accurate. That said, the Analyze Particles protocol brings many benefits to the automatic quantification of cells, as it makes a process simpler, much faster, more effective and replicable. Within the classical approaches, it is a very reliable solution to the problem presented since it was possible to prove through this that the cell counting process can be automated. To finish, this protocol will be brought to the discussion of the different methodologies for cell quantification, manual, classical and deep learning.

ITCN Runner

Another approach that automates the microglial cell counting process fetched from the results obtained with the generic cell sample experiences regarding a more classical methodology is the quantification process with the ITCN plugin. Taking advantage of the good practices gathered from the previous experiments, this approach was tested with microglial cells. Table 40 presents the results obtained in the counting process of Lobule 2 of animal CN276 2FD. Similar to the previous procedure, care was taken to

implement the best approach within the developed protocol. For this, in some images, it was necessary to test different values for the various parameters involved in the quantification process. As was already explained, before the quantification process itself, certain procedures are recommended as described in subsection 5.2.2. Taking advantage of the fact that the results obtained in the previous procedure were excellent, the pre-quantification procedures performed for each image were reused for this approach. This made the quantification process more effective.

Image	Number of Cells
1	8
2	25
3	30
4	18

Table 40: CN276 2FD - Slice 1 - ITCN - Automatic Quantification Results for Lobule 2.

Moving on to the results, the 1^{st} quantification perfomed in image one of Lobule 2 of animal CN276 2FD resulted in 8 cells quantified. The ideal cell size defined was 15 pixels, and the minimum distance between cells was 19 pixels. The 2^{nd} cell counting process resulted in 25 cells quantified, after defining the cell size in 14 pixels, and the minimum distance between cells was 39 pixels. The 3^{rd} quantification resulted in 30 cells quantified, and the 4^{th} resulted in 18microglial cells quantified. The cell size was 10 and 11 pixels respectively, and the minimum distance between cells was 55 and 47 pixels. In all counting processes, the value of the ITCN threshold was adjusted to obtain more accurate and realistic results. Despite all these precautions, as was already foreseeable and it was possible to conclude through the experiments carried out with the generic cell sample, the results were a little off from the real number of cells on the image. It should be noted that even with this performance, in some cases this process was faster and equally accurate when compared to the manual quantification process. Therefore, the remaining images were quantified to be able to substantiate with concrete results which of the classic approaches is the best alternative to the standard counting process.Tables 41, 42, 43 and 44 display the different results of the ITCN quantification process of Lobule 3, 4, 5 and 6, respectively. This quantification process is well documented which helps if it is necessary to replicate work or apply this approach in similar situations.

Image	Number of Cells
1	25
2	25
3	7
4	16
5	12

Table 41: CN276 2FD - Slice 1 - ITCN - Automatic Quantification Results for Lobule 3.

Image	Number of Cells
1	9
2	14
3	10
4	13
5	9

Table 42: CN276 2FD - Slice 1 - ITCN - Automatic Quantification Results for Lobule 4.

Image	Number of Cells
1	8
2	11
3	20
4	9
5	14

Table 43: CN276 2FD - Slice 1 - ITCN - Automatic Quantification Results for Lobule 5.

Image	Number of Cells
1	13
2	8
3	24
4	2
5	10

Table 44: CN276 2FD - Slice 1 - ITCN - Automatic Quantification Results for Lobule 6.

To conclude, the approach implemented with the ITCN plugin proved that it is possible to automate the cell counting process, namely the process of counting microglial cells. Once again, it doesn't make sense to highlight only one experience, but the whole counting process. As explained, the brain images of microglial cells are quite complex which in itself does not help in the counting process. As had already been verified with the experiments carried out with the generic cell sample, pre-quantification procedures are very important. The more rigorous they are, the more effective the counting process will be. This was no exception in the case of microglial cells. Another very relevant aspect of the quantification of cells based on this approach was the definition of cell size and the minimum distance between cells. It should be noted that not defining these two parameters correctly, the counting process is not efficient because it will either count more or fewer cells. Knowing that microglial cells are non-uniform and have different sizes and lengths from each other, sometimes it was necessary to reset cell size and the minimum distance between cells several times, which made the counting process more time-consuming. The value assigned to the ITCN threshold is again important as it helps to deal with image noise and outliers. Thus, in some experiences, to achieve results close to the actual number of cells in an image was is necessary to conduct several tests. That said, the ITCN plugin undoubtedly was able to automate the microglial cell quantification, which is one of the objectives of this dissertation. However, when we compare its

performance with the developed Analyze Particles protocol, we realize that the ITCN approach is not the ideal solution since sometimes takes longer to achieve proper results. Perhaps in situations where the cells are all of a similar size and are evenly distributed across the image, the ITCN plugin may be a better solution to the problem when compared to the developed protocol. Through the analysis of the results and the experiments carried out, it can be concluded that the ITCN works better in situations that are not too complex, which is not the case with microglial cells. Therefore, this fully documented protocol will not be brought for the discussion of the different methodologies for cell quantification, manual, classical and deep learning, as the previous approach evidence better results.

6.2 Deep Learning for Automatic Cell Counting

Based on CNNs, which is the model of choice for automatic cell quantification with a deep learning-based methodology, several experiments were performed, as was detailed in the previous chapter. Once again, tests were conducted with the generic cell sample to understand the possible limitations of the model, as well as to study the pros and cons of the application thereof. So, through the results obtained, a discussion is presented in subsection 6.2.1. The outcomes of the model made it possible to conclude that this approach can be applied in the case of microglial cells. Subsection 6.2.2 details the results obtained with microglial cells, as well as, presents a discussion, and bases the procedure considered optimal. The chosen procedure will be brought to the comparison between the different approaches, the manual, the classic and the deep learning approach.

6.2.1 Cell Colony Sample

Similar to the classical approach, this generic cell sample was selected to understand the limitations of the CNNs model. However, unlike the classical methodology, in the case of deep learning approaches, the cell sample was divided into 16 equal parts to build a dataset composed of 16 images and thus obtain more adequate and accurate results. Next, the results gathered from the experiments performed are presented. For each strategy, results are discussed, followed by the conclusions drawn that will positively influence the microglial cell counting process.

Classification per Number of Cells

One of the approaches for automating the counting process based on a deep learning methodology is image classification based on the number of cells. The classification model is composed of four parameters, namely the bath size, the number of epochs, the learning rate and whether or not to apply data augmentation. These parameters can take different values so it was necessary to tune the model as explained in subsection 5.3.1. The tuning performed on the model resulted in 330 experiments. Table 45 presents the top 15 of the best results obtained. In addition to the results, the values that the various parameters took to arrive at the presented results are also presented. As explained, to arrive at the score

value presented, cross-validation was performed with 16 folds and the final error is the average of the 16 folds. In this way, it was possible to maximize the training data since in each fold we train with 15 images and test only with 1 image.

batch_size	epochs	learning_rate	data_augmentation	score	loss	run_time	str(score_list)
8	25	0.0005	false	0.9375	0.1678	20.3901	[0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
16	30	0.0005	false	0.9375	0.1683	22.2191	[0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
4	10	0.0010	false	0.9375	0.1706	11.5472	[0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
8	15	0.0010	false	0.9375	0.1725	13.9473	[0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
4	35	0.0010	false	0.9375	0.1882	33.9451	[0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
2	10	0.0005	false	0.9375	0.1936	14.4812	[0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
16	20	0.0010	false	0.9375	0.1991	16.4319	[0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
4	15	0.0005	false	0.9375	0.2024	15.2823	[0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
2	10	0.0010	false	0.9375	0.2219	14.4046	[0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
16	20	0.0005	false	0.9375	0.2269	15.7107	[0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
4	10	0.0005	false	0.9375	0.2273	11.4160	[0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
1	10	0.0010	false	0.9375	0.2279	17.4996	[0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
2	30	0.0010	false	0.9375	0.5378	35.5997	[0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
2	35	0.0010	false	0.9375	0.5511	38.9295	[0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
8	30	0.0010	true	0.8750	0.4016	35.0041	[0.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]

Table 45: Generic Cell Sample - Deep Learning Classification Results (Number).

Turning now to a more detailed analysis, throughout the visualization of the results is clear that in none of the experiments the model was able to correctly classify the first image. The reason could be related to the image in question, as it contains 38 cells. Therefore is labelled as "Few". The fact that it has 38 cells puts it very close to the next class since it only needed to have 40 cells to be labelled as "Average". Thus, this image turns out to be a bit complicated for a first prediction of the model and consequently helps to justify the result obtained in the classification. It is important to point out that to know the result of the prediction of a certain image, an order was defined. Another relevant aspect to highlight is that in these top-of-the-best experiences, the model managed to correctly classify 15 of the 16 images, which leads to the expectation that it will have an adequate performance when applied to microglial cells. Maybe related to the fact that the dataset is composed of 16 images, data augmentation was not used in any of the top experiments.

To conclude, of all the experiments carried out the first four should be highlighted, as they had a loss value of less than 0.18 and a run time of fewer than twenty-three seconds. The results are excellent as the model just didn't get it right in predicting one image. In common among these four best experiments we

have that the value of the learning rate was 0.0005 or 0.0010. The first value is the most complex value that the model can take. This leads to the conclusion that the model will behave well in more complex situations as will be the case of microglial cells. To attest to this, the second-best experience took the most complex values possible (except for the application of data augmentation) and still managed to be one of the best results. That said, the classification approach based on the number of cells in an image brings many benefits to the automatic quantification of cells, as it makes a process that is usually time-consuming simpler, faster and more effective. Within a deep learning-based methodology, it is a reliable solution to the problem presented and therefore it will be used with microglial cells.

Classification per Area of Cells

Another approach for automating the counting process based on a deep learning methodology is image classification based on the percentage of area that the cells take in an image. Similar to the previous approach, the classification model is composed of four parameters, namely the bath size, the number of epochs, the learning rate and whether or not to apply data augmentation. Once again these parameters can take different values so it was necessary to tune the model. The entire tuning process is explained in subsection 5.3.1. The tuning performed on the model resulted in four 480 experiments. Table 46 presents the top 15 of the best results obtained. Parallel to the previous approach, cross-validation was performed with 16 folds and the final error is the average of the 16 folds.

By viewing the results, we can prove that the model behaved very well by correctly classifying 15 of the 16 images. Once again is clear that in none of the experiments the model was able to correctly classify the first image. The reason for this could also be related to the image in question, as the percentage of the area that the cells occupied in the image are 5.159%. Therefore is labelled as "Few". To be classified as "Average" the percentage of the area needed to be 5.2%. Given the proximity of values, once again we conclude that the image in question turns out to be a bit complicated for a first prediction of the model and consequently helps to justify the result obtained in the classification. Through the results obtained in the learning curves, in this case, it was relevant to go up to 95 epochs. In this way, the complexity of the model increases considerably, so the application of data augmentation becomes relevant, having even been applied in two of the fifteen best experiments. Overall, the results lead to believing that the classification approach based on the percentage of area that the cells take in an image will have an acceptable performance when applied to microglial cells.

To conclude, highlight the two best experiments, which despite having taken more complex values for the model parameters managed to have the lowest loss value, lower than 0.17. Despite the high run time, results are excellent as the model just didn't get it right in predicting one image. In the remaining thirteen cases, the model also failed to predict one image. In common these two approaches have the same value for the batch size, which is eight, the same number of epochs, which is seventy-five, and both used data augmentation. Regarding the value of the learning rate parameter, they used the most complex ones for the model. This leads to the judgment that the model will behave reasonably in more complex situations as will be the case of microglial cells. To testify to this, the best experience took data augmentation, and

batch_size	epochs	learning_rate	data_augmentation	score	loss	run_time	str(score_list)
8	75	0.0005	true	0.9375	0.1593	85.7092	[0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
8	75	0.0010	true	0.9375	0.1612	85.7127	[0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
8	65	0.0005	false	0.9375	0.5792	47.4480	[0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
8	70	0.0005	false	0.9375	0.5959	54.7171	[0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
8	65	0.0010	false	0.9375	0.7511	54.4616	[0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
8	60	0.0010	false	0.9375	0.7541	49.5269	[0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
8	70	0.0010	false	0.9375	0.8202	56.6123	[0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
8	75	0.0010	false	0.9375	0.8563	57.2924	[0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
1	65	0.0005	false	0.9375	0.8674	106.3055	[0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
8	75	0.0005	false	0.9375	0.9085	61.6134	[0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
8	80	0.0005	false	0.9375	0.9547	65.1747	[0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
8	90	0.0005	false	0.9375	0.9570	64.7027	[0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
4	60	0.0005	false	0.9375	1.0262	61.0437	[0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
1	70	0.0005	false	0.9375	1.1333	116.4780	[0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
16	95	0.0010	false	0.9375	1.6979	65.3831	[0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]

Table 46: Generic Cell Sample - Deep Learning Classification Results (Area).

the value of the learning rate was the most complex. This model also took a high value for the number of epochs and still managed the best results despite the somewhat high run time. That said, the classification approach based on the percentage of area that the cells take in an image fetches many advantages to the automatic quantification of cells, as it makes the process faster and more effective. Within a deep learning-based methodology, this is another reliable solution to the problem presented and therefore it will be used with microglial cells.

6.2.2 Brain Image Samples

Taking advantage of the division that was performed on the images within the classical approach, namely due to the fact that the microglial cells and the marker associated with them are only visible in the red channel, the same images were reused, thus composing a dataset with 661 images. With this number of images, we are guaranteed to obtain more adequate and exact results with the model predictions. Next, the results collected from the experiments performed are given. For each strategy, results are discussed, followed by the conclusions drawn that will positively influence an approach considered optimal for the automation of the microglial cell counting process.

Classification per Number of Cells

As it was possible to prove, not only through relevant literature and documentation but also through the results obtained with the generic cell sample-related experiments, an approach to automate the microglial cell quantification based on a deep learning methodology is image classification based on the number of cells. Taking advantage of the model that obtained excellent results with the cell colony sample, the model for microglia cells also has four fundamental parameters. These parameters are once again the bath size, the number of epochs, the learning rate and whether or not to apply data augmentation. As expected these parameters took different values so it was necessary to tune the model as explained in subsection 5.3.2. The tuning performed on the model resulted in 288 experiments. Table 47 presents the top 15 of the best results obtained. As can be visualized, in addition to the classification results, the values that the various parameters took to arrive at these results are also presented. As explained, to arrive at the score value presented, cross-validation was performed with 5 folds and the final error is the average of the 5 folds. In this way, it was possible to maximize the training data since in each fold we train with 529 samples and test with 132 images.

batch_size	epochs	learning_rate	data_augmentation	score	loss	run_time	str(score_list)
32	20	0.0005	false	0.9021	0.2479	326.4157	[0.5714, 0.9393, 1.0000, 1.0000, 1.0000]
32	20	0.0010	false	0.9006	0.2765	260.9422	[0.5789, 0.9318, 0.9924, 1.0000, 1.0000]
16	20	0.0005	false	0.8690	0.3291	333.8006	[0.4586, 0.8863, 1.0000, 1.0000, 1.0000]
32	15	0.0010	false	0.8672	0.3326	202.2335	[0.5864, 0.7500, 1.0000, 1.0000, 1.0000]
16	15	0.0005	false	0.8520	0.4301	293.1848	[0.6466, 0.6136, 1.0000, 1.0000, 1.0000]
32	10	0.0005	false	0.8489	0.3904	150.9446	[0.6691, 0.6363, 0.9469, 0.9924, 1.0000]
32	15	0.0005	false	0.8475	0.4282	206.0727	[0.5939, 0.6590, 0.9848, 1.0000, 1.0000]
32	10	0.0010	false	0.8475	0.3738	179.2750	[0.6240, 0.6363, 0.9848, 1.0000, 0.9924]
16	10	0.0005	false	0.7959	0.5477	156.7259	[0.6691, 0.5454, 0.7651, 1.0000, 1.0000]
8	15	0.0005	false	0.7944	0.5796	266.0632	[0.6691, 0.5530, 0.7575, 0.9924, 1.0000]
16	15	0.0010	false	0.7595	0.8250	209.3782	[0.6691, 0.8409, 0.3560, 0.9393, 0.9924]
32	5	0.0010	false	0.7462	0.6996	76.5467	[0.6090, 0.8181, 0.5151, 0.8409, 0.9469]
16	5	0.0005	false	0.6611	0.8956	93.8852	[0.6691, 0.8030, 0.3636, 0.6212, 0.8484]
32	5	0.0005	false	0.6520	0.8672	84.6684	[0.6691, 0.8333, 0.3484, 0.5378, 0.8712]
32	4	0.0005	false	0.5989	0.9619	86.9296	[0.6691, 0.8409, 0.3181, 0.4318, 0.7348]

Table 47: Microglial Cells - Deep Learning Classification Results (Number).

Looking in more detail at the results, the first eight experiments should be highlighted, since the score obtained is above 84%. In at least one of the folds, the 132 images were correctly classified, which attests to the solidity of the model for the presented problem. However, the problem with the model was the classification made in the first fold, where the results were sometimes not the best. It is important to note that the brain images of microglial cells are extremely complex, since the cells have a complex morphology, vary in size and length, and some have longer ramifications than others. This makes the model predictions more difficult. Despite this, in the second fold, we see substantial improvements, which leads to the conclusion that the model is capable of learning from errors and thus obtaining and making better predictions. Nevertheless, the loss obtained in these eight best experiments is less than 0.44 which is a very good benchmark for the model. Another relevant aspect to highlight is that in these top-of-the-best experiences, the model didn't use data augmentation despite being composed of a high number of samples. Finally, highlight again the eight best experiments, as they took some of the most complex values

for the model parameters, such as higher values for the bath size, number of epochs, and the learning rate, and still managed to obtain the best ratings.

To conclude and in this way support an optimal approach to the problem presented in this dissertation, we highlight the first and best experience. For the reasons presented above, we know that the samples that make up our dataset (images of microglial cells) are quite complex. Then, with the tuning done to the model, we bring more complexity to it, given the values that the different parameters can take. A person not familiarised with the subject would say that great results would not be expected given the difficulty we are causing to the model, through extremely complex images and various values assigned to parameters. The results obtained contradict this premise. As if that were not enough, in addition to the already mentioned difficulty caused by the complexity of the images, the values of the parameters that gave rise to the best result were the most complex possible, except for the data augmentation, which was not used. The model took the value of 32 for the batch size with 20 epochs and 0.0005 for the learning rate. With all this difficulty, the model was able to correctly classify more than 90% of the cases, which means it correctly classified 596 images out of 661. It should be noted that the classifications made in the last three folds contributed to this, where the percentage of success was 100%. This gives 396 correctly classified images. Even the results obtained in the first two folds are very positive, with the percentage of hits exceeding 57% and 93%, respectively. As if the score wasn't enough, the model was able to classify all 661 images in 5 minutes and 44 seconds. That said, the classification approach based on the number of cells in an image brings so many benefits to the automatic quantification of cells, as it makes a process simpler, much faster, more effective and replicable. Within a deep learning-based methodology, it is a very reliable solution to the problem presented.

Classification per Area of Cells

Another approach to automate the microglial cell counting process fetched from relevant literature and through the results obtained with the generic cell sample-related experiments regarding a deep learning methodology is image classification based on the percentage of area that the cells take in an image. Once again taking advantage of the model that obtained excellent results with the cell colony sample, the model for microglia cells also has the same parameters. As the parameters can take different values was necessary to tune the model. The tuning process is described in subsection 5.3.1. The tuning performed on the model resulted in 324 experiments. Table 48 presents the top 15 of the best results obtained. Parallel to the last procedure, cross-validation was performed with 5 folds and the final error is the average of the 5 folds.

Contrary to the previous approach, but as expected from the results obtained in the classification by the percentage of area that cells occupy in an image with the generic cell sample, the results were not so positive. Despite this, the seven best experiences obtained a score percentage of more than 74%. This means that in these experiments at least 493 images were correctly classified. This attests to the solidity of the approach for the presented problem. Once again, and to explain why these results were obtained, it is worth noting that the images of microglial cells are complex, and in the case of a classification approach

batch_size	epochs	learning_rate	data_augmentation	score	loss	run_time	str(score_list)
16	25	0.0005	false	0.8312	0.5005	186.4194	[0.3533, 0.8030, 1.0000, 1.0000, 1.0000]
16	20	0.0005	false	0.7827	0.7110	145.2895	[0.3533, 0.5606, 1.0000, 1.0000, 1.0000]
32	25	0.0005	false	0.7721	0.6995	181.540	[0.3759, 0.5000, 0.9848, 1.0000, 1.0000]
8	25	0.0005	false	0.7525	0.7375	250.4927	[0.3383, 0.4318, 0.9924, 1.0000, 1.0000]
32	20	0.0005	false	0.7479	0.5524	146.7203	[0.3383 0.4848, 0.9166, 1.0000, 1.0000]
8	20	0.0005	false	0.7479	0.8725	206.0794	[0.3383, 0.4242, 0.9772, 1.0000, 1.0000]
16	15	0.0005	false	0.7464	0.6019	129.4396	[0.3383, 0.4848, 0.9166, 0.9924, 1.0000]
32	15	0.0005	false	0.6919	0.6063	100.8088	[0.3383, 0.4242, 0.7121, 0.9848, 1.0000]
16	15	0.0010	false	0.6706	0.7802	109.5244	[0.3383, 0.4469, 0.5757, 0.9924, 1.0000]
32	25	0.0010	false	0.6510	0.6611	172.7800	[0.3383, 0.4393, 0.5757, 0.9090, 0.9924]
16	10	0.0005	false	0.6449	0.7910	80.5878	[0.3383, 0.4469, 0.4924, 0.9469, 1.0000]
32	20	0.0010	false	0.6313	0.7594	162.2263	[0.3383, 0.4318, 0.4924, 0.9015, 0.9924]
8	15	0.0005	false	0.6222	0.8347	151.2869	[0.3383, 0.4469, 0.4545, 0.8712, 1.0000]
32	10	0.0005	false	0.5903	0.7720	67.7552	[0.3383, 0.4318, 0.5303, 0.7803, 0.8712]
16	10	0.0010	false	0.5706	0.8302	80.6591	[0.3383, 0.4469, 0.4545, 0.6287, 0.9848]

Table 48: Microglial Cells - Deep Learning Classification Results (Area).

based on the percentage of area that cells occupy in an image it is even worse since the cells vary in size and length. Consequently, this makes the model predictions more difficult. The results obtained in the first fold are not the best, but they improve from fold to fold. This proves that despite everything the model is capable of learning from errors and accordingly making better predictions. Anyway, the loss obtained in these seven best experiments is less than 0.88 which is also a very good benchmark for the model. In all these top-of-the-best experiences, the model didn't use data augmentation which leads to the idea that, in addition to the already extremely intricate images, the use of data augmentation would only worsen the results obtained, since it would bring an additional difficulty to the model. Finally, these experiments took some of the most complex values for the model parameters, such as 32 for the batch size, 25 for the number of epochs, and 0.0005 for the learning rate value, and still managed to obtain decent results.

To support an optimal approach to the problem presented in this dissertation, we highlight the first and best experience. For the reasons presented above, we know that the samples that make up our dataset are complex. Then, with the tuning done to the model, we bring more complexity to the procedure. With all this difficulty, the model was able to correctly classify more than 83% of the cases, which means it correctly classified 548 images out of 661. The classifications made in the last three folds contributed to this, where the percentage of success was 100%. This gives 396 correctly classified images. Looking at the value obtained in the loss, which, despite not being bad, is not ideal. The classification time took 3 minutes and 10 seconds. That said, the classification approach based on the area that cells occupy in an image brings benefits to the automatic quantification of cells, as it makes a process simpler, much faster and replicable. Nevertheless, this approach will not be brought for the discussion of the different methodologies for cell quantification, as the previous procedure presents better results to automate the process of microglial cell count.

7

Conclusion

The aim of this dissertation was the automatic quantification of microglial cells from brain images with classic and deep learning methods. Our contributions were described in the previous chapters, following two main topics: classic and deep learning methods for automatic cell counting. For each approach, we have presented and described the developed procedure and the obtained results as well as presented a discussion of those results. Therefore, in this last chapter, we sum up to the reader the main contributions and conclusions. Finally, we complete this document with our perspectives for future work that may deserve further investigation.

7.1 General Conclusions

Microglia are a type of neuronal cell located throughout the brain's spinal cord. Bearing in mind the importance of these cells and knowing how they are counted, which requires the segmentation of several images, a task usually performed manually. Therefore, we focused the work in this dissertation on studying the different methodologies that help automate microglial cell quantification. As this is a fundamental procedure that in some cases may help in the detection of an illness, the ultimate objective is enabling the development of automated computerized solutions. The study carried out is very important as automatic cell quantification approaches are becoming increasingly important to reduce the workload of specialists and provide robust and reproducible results.

As stated, nowadays, most of the cell counting processes are done manually. Conventional cell counting involves a specific set of tools and devices developed for that purpose. This process is tedious, time-consuming, and inaccurate due to operator-dependent biases. As cell counting is an important procedure routine, various study reports are focusing on the experience of the development of image processing programs and techniques to automate cell counting. Consequently, this makes the cell quantification process more time-efficient and reduces error.

To automate the cell counting process emerge classic and deep learning methodologies. Within the so-called classic methodology, we have software and assistants, like ImageJ, that automate the quantification process. Needs to be pointed out that some of these programs, developed to automate the cell

counting process, require specific settings on an image to obtain reasonable accuracy. Recently, deep learning-based approaches evidenced promising performance in various image analysis tasks, such as classification. Deep learning approaches showed similar accuracy to manual counting but a significant enhancement in reproducibility, throughput efficiency and reduced error from human factors. Here, models based on CNNs emerge since regression-based cell counting avoids the challenging task of detection or segmentation.

In the following subsections, we overview our main conclusions and contributions regarding each methodology studied in this dissertation. Also presented are the reasons that lead us to conclude that an automated process for the quantification of microglial cells is more effective than conventional processes.

7.1.1 Classic Methods for Automatic Cell Counting

Within the classical methodology, two approaches were developed to automate the quantification of microglial cells. To understand the pros and cons of these procedures and define a better work strategy, firstly they were applied to a cell colony sample, which makes up a total of four approaches within the classical methodology. In section 5.2, we present the two approaches that allowed us to solve the problem of the automatic quantification of microglial cells. In section 6.1, we present the results inherent to each counting process developed. Among the two approaches, the one to highlight is the Analyze Particles protocol developed. As the objective of this dissertation was to compare among the different methodologies the best approach to automate the microglial cell counting process, the Analyze Particles protocol is capable of being faster and more effective. To be effective needs to be pointed out that in this quantification process, based on a classical methodology, some pre-quantification procedures must be applied to the images, otherwise the counting process fails to achieve adequate results. Through the results obtained, we concluded that both the pre-quantification process and the quantification itself, within the optimal approach presented, were successfully implemented. The way the entire process is documented to the reader helps any user who follows the correct procedures to obtain the same results, and if we are faced with a different problem, this approach can serve as a basis, and in this way, the entire counting process is automated.

7.1.2 Deep Learning Methods for Automatic Cell Counting

Regarding a deep learning-based methodology, once again, two different approaches were developed to automate the quantification of microglial cells. To understand the pros and cons of these approaches, they were also applied to a cell colony sample, which makes up a total of four approaches within a deep learning methodology. In section 5.3, we present the two approaches that allowed for solving the problem of automatic quantification of microglial cells. In section 6.2, we give the results inherent to the counting processes developed. Between both, and not wanting to completely disregard the classification approach based on the percentage of the area that the cells occupy in the image, the one that stands out is the

classification approach based on the number of cells contained in an image, as it was the one that had the best and fastest results. In 5 minutes and 44 seconds, the model classifies the 661 images with an accuracy percentage greater than 90%. This gives approximately 596 correctly classified images out of 661. The robustness and the results associated with the model and its ability to successfully deal with complex situations, in addition to the already expected complexity generated by the images, unquestionably attest to the benefit of applying a methodology based on deep learning to the automatic quantification of microglial cells problem. The way the entire process is documented helps the reader who follows the correct procedures to obtain the same results, and if we are faced with a different problem, this approach can serve as a basis.

7.1.3 Benefits of Automated Cell Counting Procedures

After presenting and bringing to the conclusion the ideal procedures within each of the methodologies studied, it is now up to us to compare them with the so-called common approach, manual quantification. Within the classical methodology, as expected, the optimal strategy developed and presented with ImageJ relatively easily automates the counting process of microglial cells. Thus, it is an obvious solution to the problem presented in this dissertation. Another relevant aspect that attests to the benefits of this procedure, is the fact that it can be replicated in similar situations. Regarding the deep learning-based approach, as was predictable, since these types of techniques can deal with complex problems with relative ease, the ideal procedure based on a CNNs model was able to automate the quantification of microglial cells with similar accuracy to manual approaches. Once again, this method can be replicated in similar situations. Both the Analyze Particles protocol developed and the classification based on the number of cells in an image helped to turn a process that normally is tedious, time-consuming, with high labour costs and imprecise because it is dependent on the operator into an automated process. This process is now less time-consuming, more accurate and where the error associated with users is reduced since a large part of the process is done by machine. To conclude, as one of the objectives of this dissertation is to present the best approach to quantify microglial cells, and knowing that a manual procedure is not the best solution, the choice falls on the deep learning implementation. The reason was that this procedure is extremely effective and much faster throughout the entire process. It should be noted that if a deep learning implementation had not been studied, the classical approach would have been easily chosen, as it is also a competitive solution with state-of-the-art methods.

7.2 Prospects for Future Work

New methods are constantly emerging in the clinical and medical context. Cell quantification is no exception, as it is a very important procedure routine that, in some cases, may help in the detection of a serious illness. The current and new approaches are progressively improving several tasks, namely the automation of the cell quantification process. Despite this, there is always still room for improvement. The

work developed in this dissertation is no exception, as some topics can be studied and improved in the future. There is no unique way to automate the microglial cell quantification process, as was proved in this work, by presenting two approaches for each methodology. Many other strategies may lead to better results as far as automating cell counting is concerned. That said, for future work stand some suggestions about classical and deep learning-based approaches that were not mentioned in this dissertation.

Regarding the classical methodologies, knowing and taking advantage of the fact that ImageJ is a java-based program, available on the Internet for the public domain, a plugin can be developed from scratch to help automate the microglial cell counting process. Additionally, as ImageJ was designed with an open architecture that provides extensibility via Java plugins other plugins that not only automate the counting process but also automate imaging processing can also be tested.

Regarding a deep learning-based methodology, it would be interesting to test other DL models, to understand if the models based on CNNs are the best for this type of problem, as seems to be the case when we analyze the relevant literature and documentation. However, still using CNNs model as the basis of the solution to the problem, testing other types of approaches in addition to the tested classification approach could be relevant. For example, with a detection or segmentation approach, the results obtained when the basis of the procedure is the percentage of area that the cells occupy in the image, the results could be better than when the basis of the procedure is the number of cells contained in the image.

Bibliography

[1] L. Lawson, V. Perry, and S. Gordon. “Turnover of resident microglia in the normal adult mouse brain”. In: *Neuroscience* 48.2 (1992), pp. 405–415. issn: 0306-4522. doi: https://doi.org/10.1016/0306-4522(92)90500-2 (cit. on pp. 1, 5).

[2] H. Wake et al. “Resting Microglia Directly Monitor the Functional State of Synapses In Vivo and Determine the Fate of Ischemic Terminals”. In: *Journal of Neuroscience* 29.13 (2009), pp. 3974–3980. issn: 0270-6474. doi: 10.1523/JNEUROSCI.4363-08.2009 (cit. on pp. 1, 5).

[3] H. Kettenmann et al. “Physiology of Microglia”. In: *Physiological Reviews* 91.2 (2011), pp. 461–553. doi: 10.1152/physrev.00011.2010 (cit. on pp. 1, 5–8).

[4] J. Schindelin et al. “Fiji: an open-source platform for biological-image analysis”. In: *Nature methods* 9.7 (2012). doi: https://doi.org/10.1038/nmeth.2019 (cit. on pp. 1, 18).

[5] M. S. Alkoffash et al. “A survey of digital image processing techniques in character recognition”. In: *International Journal of Computer Science and Network Security (IJCSNS)* 14.3 (2014), p. 65 (cit. on pp. 2, 12).

[6] R. C. Gonzalez, R. E. Woods, and B. R. Masters. *Digital image processing*. 2009 (cit. on pp. 2, 16).

[7] K. Kierdorf and M. Prinz. “Microglia in steady state”. In: *The Journal of Clinical Investigation* 127.9 (2017), pp. 3201–.3209. doi: 10.1172/JCI90602 (cit. on pp. 5, 8).

[8] W. Chan, S. Kohsaka, and P. Rezaie. “The origin and cell lineage of microglia—New concepts”. In: *Brain Research Reviews* 53.2 (2007), pp. 344–354. issn: 0165-0173. doi: https://doi.org/10.1016/j.brainresrev.2006.11.002 (cit. on p. 6).

[9] E. Davis, T. Foster, and W. Thomas. “Cellular forms and functions of brain microglia”. In: *Brain Research Bulletin* 34.1 (1994), pp. 73–78. issn: 0361-9230. doi: https://doi.org/10.1016/0361-9230(94)90189-9 (cit. on p. 8).

[10] J. Gehrmann, Y. Matsumoto, and G. W. Kreutzberg. “Microglia: Intrinsic immuneffector cell of the brain”. In: *Brain Research Reviews* 20.3 (1995), pp. 269–287. issn: 0165-0173. doi: https://doi.org/10.1016/0165-0173(94)00015-H (cit. on p. 8).

[11] F. Walker, M. Nilsson, and K. Jones. “Acute and Chronic Stress-Induced Disturbances of Microglial Plasticity, Phenotype and Function”. In: *Current drug targets* 14 (2013). doi: 10.2174/1389450 1113149990208 (cit. on p. 8).

[12] A. M. Jurga, M. Paleczna, and K. Z. Kuter. “Overview of General and Discriminating Markers of Differential Microglia Phenotypes”. In: *Frontiers in Cellular Neuroscience* 14 (2020), p. 198. issn: 1662-5102. doi: 10.3389/fncel.2020.00198 (cit. on p. 9).

[13] K. Ohsawa et al. “Microglia/macrophage-specific protein Iba1 binds to fimbrin and enhances its actin-bundling activity”. In: *Journal of Neurochemistry* 88.4 (2004), pp. 844–856. doi: https://doi.org/10.1046/j.1471-4159.2003.02213.x (cit. on p. 9).

[14] C. A. Glasbey and G. W. Horgan. *Image analysis for the biological sciences*. Wiley Chichester, 1995, pp. 47–184 (cit. on pp. 12, 17).

[15] A. Kaur. “A review paper on image segmentation and its various techniques in image processing”. In: *International Journal of Science And Research* (2012) (cit. on p. 13).

[16] A. A. Aly, S. B. Deris, and N. Zaki. “Research review for digital image segmentation techniques”. In: *International Journal of Computer Science & Information Technology* 3.5 (2011), p. 99 (cit. on p. 13).

[17] R. Yogamangalam and B. Karthikeyan. “Segmentation techniques comparison in image processing”. In: *International Journal of Engineering and Technology (IJET)* 5.1 (2013), pp. 307–313 (cit. on pp. 13, 14).

[18] H. G. Kaganami and Z. Beiji. “Region-Based Segmentation versus Edge Detection”. In: (2009), pp. 1217–1221. doi: 10.1109/IIH-MSP.2009.13 (cit. on pp. 13, 14).

[19] D. Comaniciu and P. Meer. “Robust analysis of feature spaces: color image segmentation”. In: (1997), pp. 750–755. doi: 10.1109/CVPR.1997.609410 (cit. on p. 14).

[20] M.-O. Baradez et al. “Robust and automated unimodal histogram thresholding and potential applications”. In: *Pattern Recognition* 37.6 (2004), pp. 1131–1148. issn: 0031-3203. doi: https://doi.org/10.1016/j.patcog.2003.12.008 (cit. on p. 14).

[21] J. Luo, R. Gray, and H.-C. Lee. “Incorporation of derivative priors in adaptive Bayesian color image segmentation”. In: (1998), pp. 780–784. doi: 10.1109/ICIP.1998.727372 (cit. on p. 14).

[22] M.-O. Baradez et al. “Robust and automated unimodal histogram thresholding and potential applications”. In: *Pattern Recognition* 37.6 (2004), pp. 1131–1148. issn: 0031-3203. doi: https://doi.org/10.1016/j.patcog.2003.12.008 (cit. on p. 14).

[23] S. Gupta, S. Mahajan, and A. K. Pandit. “A Review On Image Processing Techniques”. In: (2020), pp. 20–24. doi: 10.1109/CICN49253.2020.9242606 (cit. on pp. 15, 16).

[24] P. Liu and H. Li. “Fuzzy techniques in image restoration research-a survey”. In: *International Journal of Computational Cognition* 2.2 (2004), pp. 131–149 (cit. on p. 15).

[25] M. Maru and M. C. Parikh. “Image restoration techniques: a survey”. In: *International Journal of Computer Applications* 160.6 (2017), pp. 15–19 (cit. on p. 15).

[26] E. R. Dougherty and R. A. Lotufo. *Hands-on morphological image processing*. SPIE press, 2003, pp. 4–12 (cit. on p. 17).

[27] M. Goyal. “Morphological image processing”. In: *IJCST* 2.2 (2011) (cit. on pp. 17, 18).

[28] A. E. Carpenter et al. “CellProfiler: image analysis software for identifying and quantifying cell phenotypes”. In: *Genome biology* 7.10 (2006). doi: https://doi.org/10.1186/gb-2006-7-10-r100 (cit. on p. 18).

[29] I. V. Grishagin. “Automatic cell counting with ImageJ”. In: *Analytical Biochemistry* 473 (2015), pp. 63–65. issn: 0003-2697. doi: https://doi.org/10.1016/j.ab.2014.12.007 (cit. on p. 19).

[30] W. Burger and M. J. Burge. *Digital image processing: an algorithmic introduction using Java*. Springer, 2016, p. 566 (cit. on p. 19).

[31] T. Ferreira and W. Rasband. “ImageJ user guide”. In: *ImageJ/Fiji* 1 (2012), pp. 155–161 (cit. on p. 19).

[32] W. Bailer. “Writing ImageJ plugins—a tutorial”. In: *Upper Austria University of Applied Sciences, Austria* (2006) (cit. on p. 19).

[33] J. Schindelin et al. “The ImageJ ecosystem: An open platform for biomedical image analysis”. In: *Molecular reproduction and development* 82.7-8 (2015), pp. 518–529. doi: https://doi.org/10.1002/mrd.22489 (cit. on p. 20).

[34] K. Young and H. Morrison. “Quantifying microglia morphology from photomicrographs of immunohistochemistry prepared tissue using ImageJ”. In: *Journal of Visualized Experiments* 136 (2018). issn: 1940-087X. doi: 10.3791/57648 (cit. on p. 24).

[35] V. E. Dökümcüoğlu and M. Yılmaz. “ASSESSMENT OF CELL COUNTING METHOD BASED ON IMAGE PROCESSING FOR A MICROALGA CULTURE”. In: *Mediterranean Fisheries and Aquaculture Research* 3.2 (2020), pp. 75–81. issn: 2618-6551 (cit. on p. 25).

[36] N. F. Ash et al. “Automated segmentation and analysis of retinal microglia within ImageJ”. In: *Experimental Eye Research* 203 (2021), p. 108416. issn: 0014-4835. doi: https://doi.org/10.1016/j.exer.2020.108416 (cit. on p. 26).

[37] J. E. Tomaszewski and A. D. Ward. “Medical Imaging 2021: Digital Pathology”. In: *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*. 2021. doi: 10.1117/12.2595414 (cit. on p. 27).

[38] W. Xie, J. A. Noble, and A. Zisserman. "Microscopy cell counting and detection with fully convolutional regression networks". In: *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization* 6.3 (2018), pp. 283–292. doi: 10.1080/21681163.2016.1149104 (cit. on pp. 27, 28).

[39] Y. Xue et al. "Cell Counting by Regression Using Convolutional Neural Network". In: *Computer Vision – ECCV 2016 Workshops*. Cham: Springer International Publishing, 2016, pp. 274–290. isbn: 978-3-319-46604-0. doi: https://doi.org/10.1007/978-3-319-46604-0_20 (cit. on p. 28).

[40] W. Ertel. *Introduction to artificial intelligence*. Springer, 2018, pp. 1–3 (cit. on p. 28).

[41] J. M. Górriz et al. "Artificial intelligence within the interplay between natural and artificial computation: Advances in data science, trends and applications". In: *Neurocomputing* 410 (2020), pp. 237–270. issn: 0925-2312. doi: https://doi.org/10.1016/j.neucom.2020.05.078 (cit. on p. 28).

[42] D. Carneiro et al. "Online dispute resolution: an artificial intelligence perspective". In: *Artificial Intelligence Review* 41.2 (2014), pp. 211–240. doi: https://doi.org/10.1007/s10462-011-9305-z (cit. on p. 28).

[43] B. Fernandes, P. Novais, and C. Analide. "A Multi-Agent System for Automated Machine Learning". In: *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*. 2022, pp. 1899–1901 (cit. on p. 28).

[44] J. P. Mueller and L. Massaron. *Machine learning for dummies*. John Wiley & Sons, 2021, pp. 4–18 (cit. on pp. 28, 29, 34, 35).

[45] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016, pp. 96–119 (cit. on p. 28).

[46] B. Fernandes et al. "Long short-term memory networks for traffic flow forecasting: exploring input variables, time frames and multi-step approaches". In: *Informatica* 31.4 (2020), pp. 723–749. doi: 10.15388/20-INFOR431 (cit. on p. 29).

[47] S. Singh. *Cousins of Artificial Intelligence*. url: https://towardsdatascience.com/cousins-of-artificial-intelligence-dda4edc27b55 (visited on 12/19/2021) (cit. on p. 29).

[48] P. Oliveira et al. "Forecasting Energy Consumption of Wastewater Treatment Plants with a Transfer Learning Approach for Sustainable Cities". In: *Electronics* 10.10 (2021), p. 1149. issn: 2079-9292. doi: 10.3390/electronics10101149 (cit. on p. 29).

[49] N. Buduma and N. Locascio. *Fundamentals of deep learning: Designing next-generation machine intelligence algorithms*. O'Reilly Media, Inc., 2017, pp. 1–39. isbn: 978-1-491-92561-4 (cit. on pp. 29, 31–33).

[50] A. Karpathy and J. Johnson. *Convolutional neural networks for visual recognition*. url: https://cs231n.github.io/ (visited on 12/28/2021) (cit. on pp. 29–34).

[51] T. Dettmers. *Deep Learning in a Nutshell: Core Concepts*. url: https://developer.nvidia.com/blog/deep-learning-nutshell-core-concepts/ (visited on 01/07/2022) (cit. on pp. 32, 33).

[52] U. Karn. *An intuitive explanation of convolutional neural networks*. url: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/ (visited on 01/09/2022) (cit. on p. 33).

[53] M. Peemen et al. "The neuro vector engine: Flexibility to improve convolutional net efficiency for wearable vision". In: *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*. 2016, pp. 1604–1609. isbn: 978-3-9815-3707-9 (cit. on pp. 33, 34).

[54] A. Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, Inc., 2019, pp. 7–14. isbn: 978-1-492-03264-9 (cit. on p. 34).

[55] D. Parikh. *Learning Paradigms in Machine Learning*. url: https://medium.datadriveninvestor.com/learning-paradigms-in-machine-learning-146ebf8b5943 (visited on 11/23/2021) (cit. on pp. 34, 35).

[56] Z. Pan et al. "Recent Progress on Generative Adversarial Networks (GANs): A Survey". In: *SPSS inc* 7 (2019), pp. 36322–36333. doi: 10.1109/ACCESS.2019.2905015 (cit. on pp. 36, 37).

[57] E. Kan. *What The Heck Are VAE-GANs?* url: https://towardsdatascience.com/what-the-heck-are-vae-gans-17b86023588a (visited on 11/30/2021) (cit. on pp. 36, 37).

[58] H. Ahmady Phoulady et al. "Automatic ground truth for deep learning stereology of immunostained neurons and microglia in mouse neocortex". In: *Journal of Chemical Neuroanatomy* 98 (2019), pp. 1–7. issn: 0891-0618. doi: https://doi.org/10.1016/j.jchemneu.2019.02.006 (cit. on p. 37).

[59] P. Dave et al. "An adaptive digital stain separation method for deep learning-based automatic cell profile counts". In: *Journal of Neuroscience Methods* 354 (2021), p. 109102. issn: 0165-0270. doi: https://doi.org/10.1016/j.jneumeth.2021.109102 (cit. on p. 38).

[60] A. B. Campos et al. "Profiling Microglia in a Mouse Model of Machado-Joseph Disease". In: *Biomedicines* 10.2 (2022). issn: 2227-9059. doi: 10.3390/biomedicines10020237 (cit. on pp. 47, 48).

[61] "Implications of functional anatomy on information processing in the deep cerebellar nuclei". In: *Frontiers in Cellular Neuroscience* 3 (2009). issn: 1662-5102. doi: 10.3389/neuro.03.014.2009 (cit. on p. 47).

A

CN276 2FD Quantification Settings

This appendix displays the quantification settings applied to the brain image of the animal CN276 2FD, allowing us to achieve the results obtained in microglia cell counts with ImageJ regarding a more classical approach.

		1^{st}	2^{nd}	3^{rd}	4^{th}	5^{th}	6^{th}	7^{th}	8^{th}
Threshold	Min	0	0	0	0	0	0	0	0
	Max	3694	2168	2393	2569	2425	2794	2650	2425
		96.29%	95.58%	95.80%	97.14%	96.89%	96.39%	95.97%	96.51%
	Algorithm	Default	Default	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 49: CN276 2FD - Slice 1 - Quantification Settings for DCN.

		1^{st}	2^{nd}	3^{rd}	4^{th}
Threshold	Min	0	0	0	0
	Max	2746	3212	2875	2248
		96.32%	96.35%	95.63%	95.07%
	Algorithm	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 50: CN276 2FD - Slice 1 - Quantification Settings for Lobule 7.

		1^{st}	2^{nd}	3^{rd}	4^{th}	5^{th}	6^{th}	7^{th}
Threshold	Min	0	0	0	0	0	0	0
	Max	3003	2682	2618	2425	2232	1847	1718
		96.14%	96.11%	95.73%	95.88%	94.77%	95.87%	94.78%
	Algorithm	Default	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 51: CN276 2FD - Slice 1 - Quantification Settings for Lobule 8.

		1	2	3	4	5	6	7	8	9
Threshold	Min	0	0	0	0	0	0	0	0	0
	Max	1975	2618	1734	1799	2168	1959	2489	2216	2088
		95.60%	95.22%	94.31%	95.02%	94.63%	95.33%	94.88%	94.68%	94.91%
	Algorithm	Default	Default	Default	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 52: CN276 2FD - Slice 1 - Quantification Settings for Lobule 9.

		1	2	3	4	5	6	7	8	9
Threshold	Min	0	0	0	0	0	0	0	0	0
	Max	3404	3726	1815	2923	2409	2746	3115	3228	1927
		96.43%	95.60%	94.91%	96.15%	95.29%	96.86%	97.41%	97.41%	94.90%
	Algorithm	Default	Default	Default	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 53: CN276 2FD - Slice 1 - Quantification Settings for Lobule 10.

		1	2	3	4	5
Threshold	Min	0	0	0	0	0
	Max	2521	2473	3501	3453	3131
		95.08%	95.66%	94.49%	93.44%	94.30%
	Algorithm	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 54: CN276 2FD - Slice 1 - Quantification Settings for Lobule 11.

		1	2	3	4	5	6	7	8
Threshold	Min	0	0	0	0	0	0	0	0
	Max	1638	1092	1574	1847	1156	1686	1847	1028
		95.26%	95.22%	94.02%	93.68%	92.83%	91.90%	94.43%	90.87%
	Algorithm	Default	Default	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 55: CN276 2FD - Slice 2 - Quantification Settings for DCN.

		1	2	3	4	5	6	7	8
Threshold	Min	0	0	0	0	0	0	0	0
	Max	3437	2746	3019	2521	1750	1831	2826	2650
		96.98%	95.91%	94.87%	94.11%	94.35%	97.27%	95.87%	97.34%
	Algorithm	Default	Default	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 56: CN276 2FD - Slice 2 - Quantification Settings for Lobule 2.

		1	2	3	4	5
Threshold	Min	0	0	0	0	0
	Max	2088	2007	2312	2682	2971
		94.15%	94.13%	96.21%	93.54%	94.93%
	Algorithm	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 57: CN276 2FD - Slice 2 - Quantification Settings for Lobule 3.

		1	2	3	4	5	6
Threshold	Min	0	0	0	0	0	0
	Max	2505	2810	2569	2312	2393	3292
		93.61%	95.43%	95.45%	95.84%	96.10%	96.05%
	Algorithm	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 58: CN276 2FD - Slice 2 - Quantification Settings for Lobule 4.

		1	2	3
Threshold	Min	0	0	0
	Max	1670	1269	1702
		94.02%	93.61%	94.23%
	Algorithm	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00

Table 59: CN276 2FD - Slice 2 - Quantification Settings for Lobule 5.

		1	2	3	4	5	6
Threshold	Min	0	0	0	0	0	0
	Max	1879	2120	1590	2023	2377	2602
		93.92%	95.27%	94.21%	95.37%	95.97%	95.85%
	Algorithm	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 60: CN276 2FD - Slice 2 - Quantification Settings for Lobule 6.

		1	2	3	4	5
Threshold	Min	0	0	0	0	0
	Max	1493	996	1237	1429	1381
		95.25%	94.05%	96.21%	95.38%	95.62%
	Algorithm	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 61: CN276 2FD - Slice 2 - Quantification Settings for Lobule 7.

		1	2	3	4	5	6	7
Threshold	Min	0	0	0	0	0	0	0
	Max	1686	2264	1911	1815	1991	1574	1188
		95.50%	96.94%	96.05%	96.15%	96.54%	95.21%	95.24%
	Algorithm	Default	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 62: CN276 2FD - Slice 2 - Quantification Settings for Lobule 8.

		1	2	3	4	5	6	7
Threshold	Min	0	0	0	0	0	0	0
	Max	1847	2361	1975	1526	1638	1558	1815
		95.32%	95.72%	95.18%	95.79%	95.03%	94.35%	93.65%
	Algorithm	Default	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 63: CN276 2FD - Slice 2 - Quantification Settings for Lobule 9.

		1	2	3	4	5	6	7
Threshold	Min	0	0	0	0	0	0	0
	Max	1028	1558	1542	1702	1542	1606	1574
		97.37%	94.30%	95.29%	93.51%	94.00%	94.48%	95.95%
	Algorithm	Default	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 64: CN276 2FD - Slice 2 - Quantification Settings for Lobule 10.

		1	2	3	4	5	6	7	8	9
Threshold	Min	0	0	0	0	0	0	0	0	0
	Max	1237	1253	1269	530	658	1172	1108	1140	835
		97.71%	97.02%	97.46%	96.80%	95.05%	94.86%	93.07%	96.42%	93.39%
	Algorithm	Default	Default	Default	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 65: CN276 2FD - Slice 2 - Quantification Settings for Lobule 11.

		1	2	3	4	5	6
Threshold	Min	0	0	0	0	0	0
	Max	1349	1574	1461	1718	2184	1445
		93.09%	92.66%	95.08%	96.06%	95.51%	94.07%
	Algorithm	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 66: CN276 2FD - Slice 2 - Quantification Settings for Lobule 12.

		1	2	3	4	5
Threshold	Min	0	0	0	0	0
	Max	1204	1670	1975	1734	1645
		95.12%	96.20%	96.24%	96.35%	95.97%
	Algorithm	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 67: CN276 2FD - Slice 2 - Quantification Settings for Lobule 13.

		1	2	3	4	5	6	7
Threshold	Min	0	0	0	0	0	0	0
	Max	1879	1526	1766	1076	1959	2232	1911
		97.03%	96.20%	97.11%	98.72%	94.09%	95.40%	96.10%
	Algorithm	Default	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 68: CN276 2FD - Slice 2 - Quantification Settings for Lobule 14.

		1	2	3	4	5
Threshold	Min	0	0	0	0	0
	Max	2104	2120	1381	2361	2505
		94.36%	97.08%	98.30%	96.88%	96.91%
	Algorithm	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 69: CN276 2FD - Slice 2 - Quantification Settings for Lobule 15.

		1	2	3	4
Threshold	Min	0	0	0	0
	Max	2521	2296	2264	2104
		96.00%	96.55%	95.25%	95.96%
	Algorithm	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 70: CN276 2FD - Slice 2 - Quantification Settings for Lobule 16.

		1	2	3	4	5	6	7	8
Threshold	Min	0	0	0	0	0	0	0	0
	Max	1622	1429	1124	1365	1237	915	1204	1124
		96.18%	97.49%	97.60%	94.00%	94.37%	96.97%	95.49%	93.74%
	Algorithm	Default	Default	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 71: CN276 2FD - Slice 2 - Quantification Settings for Lobule 17.

		1	2	3	4	5	6	7	8	9	10
Threshold	Min	0	0	0	0	0	0	0	0	0	0
	Max	434	1220	1237	1188	674	418	947	1012	1028	771
		97.19%	96.57%	96.21%	97.13%	97.01%	96.92%	96.64%	94.08%	95.23%	95.78%
	Algorithm	Default	Default	Default	Default	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 72: CN276 2FD - Slice 2 - Quantification Settings for Lobule 18.

		1	2	3	4	5	6	7	8
Threshold	Min	0	0	0	0	0	0	0	0
	Max	1365	1590	867	1028	867	1012	626	803
		94.00%	94.18%	92.58%	92.91%	92.34%	94.84%	92.46%	90.27%
	Algorithm	Default	Default	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 73: CN276 2FD - Slice 3 - Quantification Settings for DCN.

		1	2
Threshold	Min	0	0
	Max	771	755
		87.10%	87.77%
	Algorithm	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00

Table 74: CN276 2FD - Slice 3 - Quantification Settings for Lobule 2.

		1	2	3	4	5	6	7	8	9	10
Threshold	Min	0	0	0	0	0	0	0	0	0	0
	Max	771	594	466	931	707	819	1108	835	899	1060
		96.46%	96.22%	94.61%	95.13%	94.89%	92.39%	94.27%	96.99%	93.86%	94.15%
	Algorithm	Default	Default	Default	Default	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 75: CN276 2FD - Slice 3 - Quantification Settings for Lobule 3.

		1	2	3	4	5	6	7	8	9	10	11	12
Threshold	Min	0	0	0	0	0	0	0	0	0	0	0	0
	Max	353	1204	1333	1574	1815	964	1253	1188	1012	1493	835	1333
		96.14%	97.12%	96.28%	95.85%	96.89%	97.83%	98.02%	96.71%	97.10%	97.08%	96.85%	97.04%
	Algorithm	Default	Default	Default	Default	Default	Default	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 76: CN276 2FD - Slice 3 - Quantification Settings for Lobule 4.

		1	2	3	4	5	6
Threshold	Min	0	0	0	0	0	0
	Max	1156	1702	1895	1461	1445	1622
		96.64%	96.79%	96.69%	96.45%	96.53%	96.29%
	Algorithm	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 77: CN276 2FD - Slice 3 - Quantification Settings for Lobule 5.

		1	2	3	4	5
Threshold	Min	0	0	0	0	0
	Max	1028	1317	1285	1542	1590
		95.54%	95.37%	93.00%	95.29%	92.67%
	Algorithm	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 78: CN276 2FD - Slice 3 - Quantification Settings for Lobule 6.

		1	2	3	4	5	6	7	8	9
Threshold	Min	0	0	0	0	0	0	0	0	0
	Max	1510	1156	883	1285	947	1285	1333	1140	1526
		96.86%	95.60%	93.19%	94.11%	92.94%	95.18%	94.53%	95.61%	93.98%
	Algorithm	Default	Default	Default	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 79: CN276 2FD - Slice 3 - Quantification Settings for Lobule 7.

		1	2	3	4	5	6	7	8	9	10	11	12	13	14
Threshold	Min	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Max	1028	1188	899	1060	1686	1349	1381	1526	1429	1477	1253	1365	1381	947
		95.72%	97.02%	95.75%	96.83%	96.64%	96.46%	96.18%	96.67%	96.67%	95.47%	96.17%	95.85%	96.44%	95.96%
	Algorithm	Default	Default	Default	Default	Default	Default	Default	Default	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 80: CN276 2FD - Slice 3 - Quantification Settings for Lobule 8.

		1	2	3	4	5	6	7	8	9
Threshold	Min	0	0	0	0	0	0	0	0	0
	Max	1140	1510	1285	899	787	1381	1204	1188	610
		96.93%	95.20%	94.46%	94.34%	94.93%	96.04%	95.30%	94.54%	95.31%
	Algorithm	Default	Default	Default	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 81: CN276 2FD - Slice 3 - Quantification Settings for Lobule 9.

		1	2	3	4	5	6	7	8	9	10
Threshold	Min	0	0	0	0	0	0	0	0	0	0
	Max	883	1028	1285	1493	947	1461	1381	1285	1301	1285
		95.15%	95.20%	96.12%	97.06%	95.88%	94.86%	92.88%	92.80%	95.45%	97.21%
	Algorithm	Default	Default	Default	Default	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 82: CN276 2FD - Slice 3 - Quantification Settings for Lobule 10.

		1	2	3	4	5	6	7	8
Threshold	Min	0	0	0	0	0	0	0	0
	Max	1220	867	964	1237	1429	1654	1317	1317
		96.62%	92.84%	98.06%	96.16%	95.95%	93.89%	95.60%	94.70%
	Algorithm	Default	Default	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 83: CN276 2FD - Slice 3 - Quantification Settings for Lobule 11.

		1	2	3	4	5	6	7
Threshold	Min	0	0	0	0	0	0	0
	Max	1253	1076	1269	1028	1156	418	530
		97.07%	95.09%	96.89%	95.13%	96.36%	96.67%	94.75%
	Algorithm	Default	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 84: CN276 2FD - Slice 3 - Quantification Settings for Lobule 12.

		1	2	3	4
Threshold	Min	0	0	0	0
	Max	787	771	1140	1445
		96.12%	96.78%	96.49%	95.30%
	Algorithm	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 85: CN276 2FD - Slice 3 - Quantification Settings for Lobule 13.

B

CN282 2TE Quantification Settings

This appendix displays the quantification settings applied to the brain image of the animal CN282 2TE, allowing us to achieve the results obtained in microglia cell counts with ImageJ regarding a more classical approach.

		1	2	3	4	5	6
Threshold	Min	0	0	0	0	0	0
	Max	3051	3324	2553	3228	4031	3421
		97.05%	97.10%	96.90%	97.25%	97.90%	98.05%
	Algorithm	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 86: CN282 2TE - Slice 1 - Quantification Settings for DCN.

		1	2	3	4	5	6	7	8
Threshold	Min	0	0	0	0	0	0	0	0
	Max	2505	2521	2361	2473	1783	1991	1959	2007
		97.65%	97.65%	97.20%	97.65%	97.65%	97.15%	97.60%	96.90%
	Algorithm	Default	Default	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 87: CN282 2TE - Slice 1 - Quantification Settings for Lobule 2.

		1	2	3	4	5
Threshold	Min	0	0	0	0	0
	Max	1927	2216	2345	1927	3003
		98.05%	97.30%	97.20%	97.20%	97.50%
	Algorithm	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 88: CN282 2TE - Slice 1 - Quantification Settings for Lobule 3.

		1	2	3	4	5	6	7	8
Threshold	Min	0	0	0	0	0	0	0	0
	Max	2088	2312	2505	2056	2023	1911	2007	2128
		98.00%	97.60%	97.84%	97.65%	97.89%	97.90%	97.81%	97.81%
	Algorithm	Default	Default	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 89: CN282 2TE - Slice 1 - Quantification Settings for Lobule 4.

		1	2	3	4	5	6	7	8
Threshold	Min	0	0	0	0	0	0	0	0
	Max	1686	2152	2312	1959	2312	1927	1590	2425
		98.09%	97.95%	98.17%	97.70%	98.23%	98.15%	97.68%	98.17%
	Algorithm	Default	Default	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 90: CN282 2TE - Slice 1 - Quantification Settings for Lobule 5.

		1	2	3	4	5	6
Threshold	Min	0	0	0	0	0	0
	Max	1959	2013	2184	1863	2056	2007
		97.70%	97.81%	98.07%	98.09%	97.72%	97.47%
	Algorithm	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 91: CN282 2TE - Slice 1 - Quantification Settings for Lobule 6.

		1	2	3
Threshold	Min	0	0	0
	Max	2007	1863	1606
		97.33%	97.93%	96.70%
	Algorithm	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00

Table 92: CN282 2TE - Slice 1 - Quantification Settings for Lobule 7.

		1	2	3
Threshold	Min	0	0	0
	Max	2505	2232	2007
		98.35%	97.99%	97.69%
	Algorithm	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00

Table 93: CN282 2TE - Slice 1 - Quantification Settings for Lobule 8.

		1	2	3	4	5	6	7
Threshold	Min	0	0	0	0	0	0	0
	Max	2826	2778	3196	2826	2939	3083	1429
		96.65%	97.01%	97.21%	98.02%	97.23%	97.84%	97.87%
	Algorithm	Default	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 94: CN282 2TE - Slice 2 - Quantification Settings for DCN.

		1	2	3	4	5	6	7	8	9
Threshold	Min	0	0	0	0	0	0	0	0	0
	Max	2023	1815	1606	2585	2136	1927	2393	1477	1445
		97.90%	97.65%	97.52%	97.86%	98.46%	98.04%	98.31%	97.82%	97.58%
	Algorithm	Default	Default	Default	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 95: CN282 2TE - Slice 2 - Quantification Settings for Lobule 2.

		1	2	3	4	5	6	7	8
Threshold	Min	0	0	0	0	0	0	0	0
	Max	1349	1542	1429	1815	1911	1783	1622	1493
		97.94%	98.34%	97.30%	97.32%	98.01%	97.71%	97.22%	97.34%
	Algorithm	Default	Default	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 96: CN282 2TE - Slice 2 - Quantification Settings for Lobule 3.

		1	2	3	4	5	6	7	8	9
Threshold	Min	0	0	0	0	0	0	0	0	0
	Max	1301	2039	1622	1783	1590	1493	1237	1140	1285
		97.77%	97.99%	97.77%	97.94%	97.58%	97.41%	98.41%	98.52%	97.85%
	Algorithm	Default	Default	Default	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 97: CN282 2TE - Slice 2 - Quantification Settings for Lobule 4.

		1	2	3	4	5	6	7	8	9
Threshold	Min	0	0	0	0	0	0	0	0	0
	Max	1895	1975	2023	1783	1927	1831	2088	2411	2088
		98.71%	98.42%	98.07%	98.18%	97.71%	98.21%	98.15%	98.47%	98.98%
	Algorithm	Default	Default	Default	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 98: CN282 2TE - Slice 2 - Quantification Settings for Lobule 5.

		1	2	3	4	5	6	7	8	9
Threshold	Min	0	0	0	0	0	0	0	0	0
	Max	1783	1718	1477	1975	1783	1574	1493	1317	1799
		98.16%	97.47%	97.97%	98.18%	98.33%	97.30%	98.19%	97.86%	97.72%
	Algorithm	Default	Default	Default	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 99: CN282 2TE - Slice 2 - Quantification Settings for Lobule 6.

		1	2	3	4	5
Threshold	Min	0	0	0	0	0
	Max	1622	915	1413	1911	1766
		97.87%	99.38%	98.93%	98.23%	98.31%
	Algorithm	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 100: CN282 2TE - Slice 2 - Quantification Settings for Lobule 7.

		1	2	3	4	5	6	7
Threshold	Min	0	0	0	0	0	0	0
	Max	2987	2457	3292	2746	2666	2714	2810
		96.70%	96.84%	96.99%	96.15%	96.91%	95.74%	96.55%
	Algorithm	Default	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 101: CN282 2TE - Slice 4 - Quantification Settings for DCN.

		1	2	3	4	5	6	7	8	9
Threshold	Min	0	0	0	0	0	0	0	0	0
	Max	2248	2248	2312	1927	1831	2120	1893	2184	1799
		97.65%	97.14%	96.96%	97.32%	98.04%	97.14%	96.37%	97.83%	96.79%
	Algorithm	Default	Default	Default	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 102: CN282 2TE - Slice 4 - Quantification Settings for Lobule 2.

		1	2	3	4	5	6	7
Threshold	Min	0	0	0	0	0	0	0
	Max	1975	2361	2409	2810	2200	2345	2007
		97.98%	97.52%	97.81%	98.04%	97.98%	97.35%	97.22%
	Algorithm	Default	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 103: CN282 2TE - Slice 4 - Quantification Settings for Lobule 3.

		1	2	3	4	5	6	7	8	9	10	11
Threshold	Min	0	0	0	0	0	0	0	0	0	0	0
	Max	2120	2296	2698	2361	2200	2088	3035	2296	2473	2280	2425
		97.94%	98.54%	97.97%	98.12%	98.27%	98.64%	98.52%	98.32%	98.42%	98.12%	97.97%
	Algorithm	Default	Default	Default	Default	Default	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 104: CN282 2TE - Slice 4 - Quantification Settings for Lobule 4.

		1	2	3	4	5	6	7	8	9	10
Threshold	Min	0	0	0	0	0	0	0	0	0	0
	Max	2136	2264	2425	2489	2393	1975	2585	2746	2875	2666
		98.32%	98.72%	98.77%	98.58%	99.48%	99.38%	98.77%	98.40%	98.67%	98.77%
	Algorithm	Default	Default	Default	Default	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 105: CN282 2TE - Slice 4 - Quantification Settings for Lobule 5.

		1	2	3	4	5	6
Threshold	Min	0	0	0	0	0	0
	Max	2232	1429	2216	1510	2409	1622
		97.96%	97.71%	97.43%	97.17%	97.78%	96.81%
	Algorithm	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 106: CN282 2TE - Slice 4 - Quantification Settings for Lobule 6.

		1	2	3	4	5	6	7
Threshold	Min	0	0	0	0	0	0	0
	Max	2618	2280	2106	1317	1927	1638	2858
		97.59%	97.35%	97.94%	97.86%	97.54%	98.17%	97.91%
	Algorithm	Default	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 107: CN282 2TE - Slice 4 - Quantification Settings for Lobule 7.

		1	2	3
Threshold	Min	0	0	0
	Max	2168	2280	1734
		97.87%	97.70%	97.35%
	Algorithm	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00

Table 108: CN282 2TE - Slice 4 - Quantification Settings for Lobule 8.

C

CN283 2FD Quantification Settings

This appendix displays the quantification settings applied to the brain image of the animal CN283 2FD, allowing us to achieve the results obtained in microglia cell counts with ImageJ regarding a more classical approach.

		1	2	3	4	5	6
Threshold	Min	0	0	0	0	0	0
	Max	1445	1943	2200	2312	2248	2473
		98.04%	98.35%	98.20%	97.95%	98.10%	98.45%
	Algorithm	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 109: CN283 2FD - Slice 1 - Quantification Settings for DCN.

		1	2	3	4	5	6	7	8	9	10
Threshold	Min	0	0	0	0	0	0	0	0	0	0
	Max	1590	1108	1895	1718	1734	1638	1558	1349	1188	787
		98.55%	98.75%	98.90%	98.25%	99.00%	99.30%	99.05%	99.25%	99.45%	98.55%
	Algorithm	Default	Default	Default	Default	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 110: CN283 2FD - Slice 1 - Quantification Settings for Lobule 2.

		1	2	3	4	5	6	7	8
Threshold	Min	0	0	0	0	0	0	0	0
	Max	1493	2136	1975	1879	1477	2296	2393	3340
		98.33%	98.00%	98.10%	98.00%	99.60%	98.90%	98.50%	98.95%
	Algorithm	Default	Default	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 111: CN283 2FD - Slice 1 - Quantification Settings for Lobule 3.

		1	2	3	4	5	6	7
Threshold	Min	0	0	0	0	0	0	0
	Max	3276	2939	3356	1879	2425	1493	2875
		97.95%	98.35%	98.70%	97.95%	98.95%	98.60%	98.20%
	Algorithm	Default	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 112: CN283 2FD - Slice 1 - Quantification Settings for Lobule 4.

		1	2	3	4	5	6	7
Threshold	Min	0	0	0	0	0	0	0
	Max	2585	2457	2955	2425	2489	2425	1750
		98.40%	97.62%	98.80%	98.50%	98.65%	98.80%	99.20%
	Algorithm	Default	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 113: CN283 2FD - Slice 1 - Quantification Settings for Lobule 5.

		1	2	3	4	5	6	7
Threshold	Min	0	0	0	0	0	0	0
	Max	2409	2585	1799	2072	1783	2698	2746
		98.50%	98.75%	99.20%	98.50%	97.90%	98.60%	99.20%
	Algorithm	Default	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 114: CN283 2FD - Slice 1 - Quantification Settings for Lobule 6.

		1	2	3	4	5	6	7	8
Threshold	Min	0	0	0	0	0	0	0	0
	Max	1927	1991	1526	1413	1172	1220	1044	1445
		98.50%	98.30%	98.25%	98.60%	98.90%	98.55%	98.00%	98.00%
	Algorithm	Default	Default	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 115: CN283 2FD - Slice 1 - Quantification Settings for Lobule 7.

		1	2	3	4	5	6
Threshold	Min	0	0	0	0	0	0
	Max	2071	1959	2361	2184	2312	1542
		97.90%	98.25%	98.25%	98.85%	98.25%	98.50%
	Algorithm	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 116: CN283 2FD - Slice 4 - Quantification Settings for DCN.

		1	2	3	4	5
Threshold	Min	0	0	0	0	0
	Max	1493	1702	1718	1879	1702
		98.55%	98.45%	98.60%	98.70%	98.50%
	Algorithm	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 117: CN283 2FD - Slice 4 - Quantification Settings for Lobule 2.

		1	2	3	4
Threshold	Min	0	0	0	0
	Max	1477	1696	1702	1911
		98.05%	97.80%	97.30%	98.05%
	Algorithm	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 118: CN283 2FD - Slice 4 - Quantification Settings for Lobule 3.

		1	2	3	4	5	6
Threshold	Min	0	0	0	0	0	0
	Max	1895	1911	2120	2345	2232	3164
		98.50%	97.75%	98.50%	98.46%	98.90%	98.90%
	Algorithm	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 119: CN283 2FD - Slice 4 - Quantification Settings for Lobule 4.

		1	2	3	4	5	6
Threshold	Min	0	0	0	0	0	0
	Max	1702	1959	1574	1510	1349	1349
		98.50%	98.30%	98.95%	98.03%	97.50%	97.95%
	Algorithm	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 120: CN283 2FD - Slice 4 - Quantification Settings for Lobule 5.

		1	2	3	4	5	6
Threshold	Min	0	0	0	0	0	0
	Max	1349	2634	2296	1429	1397	1638
		98.40%	98.60%	98.95%	98.70%	98.30%	98.70%
	Algorithm	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 121: CN283 2FD - Slice 4 - Quantification Settings for Lobule 6.

		1	2	3	4
Threshold	Min	0	0	0	0
	Max	1493	1493	1606	1381
		98.25%	98.51%	98.30%	99.00%
	Algorithm	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 122: CN283 2FD - Slice 4 - Quantification Settings for Lobule 7.

		1	2	3
Threshold	Min	0	0	0
	Max	1172	1108	1188
		98.75%	98.80%	98.35%
	Algorithm	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00

Table 123: CN283 2FD - Slice 4 - Quantification Settings for Lobule 8.

D

CN284 TDTE Quantification Settings

This appendix displays the quantification settings applied to the brain image of the animal CN284 TDTE, allowing us to achieve the results obtained in microglia cell counts with ImageJ regarding a more classical approach.

		1	2	3	4	5	6	7
Threshold	Min	0	0	0	0	0	0	0
	Max	1044	1220	980	1060	1060	1092	964
		95.15%	96.08%	97.68%	95.20%	94.12%	96.23%	96.22%
	Algorithm	Default	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 124: CN284 TDTE - Slice 1 - Quantification Settings for DCN.

		1	2	3	4	5	6	7
Threshold	Min	0	0	0	0	0	0	0
	Max	498	674	691	835	658	723	674
		96.05%	96.62%	95.96%	94.93%	95.94%	94.83%	95.11%
	Algorithm	Default	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 125: CN284 TDTE - Slice 1 - Quantification Settings for Lobule 2.

		1	2	3	4	5	6
Threshold	Min	0	0	0	0	0	0
	Max	1044	915	947	835	1349	691
		96.79%	96.02%	95.65%	97.53%	97.52%	96.59%
	Algorithm	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 126: CN284 TDTE - Slice 1 - Quantification Settings for Lobule 3.

		1	2	3	4	5	6	7
Threshold	Min	0	0	0	0	0	0	0
	Max	996	1172	964	947	1076	1188	851
		97.43%	97.22%	95.04%	96.99%	91.53%	97.09%	95.71%
	Algorithm	Default	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 127: CN284 TDTE - Slice 1 - Quantification Settings for Lobule 4.

		1	2	3	4	5	6
Threshold	Min	0	0	0	0	0	0
	Max	1188	1092	1397	996	1327	1461
		97.09%	96.35%	95.93%	95.91%	97.27%	97.22%
	Algorithm	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 128: CN284 TDTE - Slice 1 - Quantification Settings for Lobule 5.

		1	2	3	4	5	6
Threshold	Min	0	0	0	0	0	0
	Max	1253	1012	787	819	1108	1172
		95.85%	94.56%	94.10%	94.22%	94.21%	97.43%
	Algorithm	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 129: CN284 TDTE - Slice 1 - Quantification Settings for Lobule 6.

		1	2	3	4	5	6	7	8
Threshold	Min	0	0	0	0	0	0	0	0
	Max	803	771	1253	1461	1012	899	1060	1188
		97.14%	95.70%	95.85%	96.63%	96.83%	94.46%	95.98%	97.26%
	Algorithm	Default	Default	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 130: CN284 TDTE - Slice 1 - Quantification Settings for Lobule 7.

		1	2	3	4	5
Threshold	Min	0	0	0	0	0
	Max	1012	1012	642	915	899
		95.57%	95.81%	90.61%	91.85%	94.82%
	Algorithm	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 131: CN284 TDTE - Slice 2 - Quantification Settings for DCN.

		1	2	3	4
Threshold	Min	0	0	0	0
	Max	851	530	626	707
		96.04%	94.74%	94.14%	93.06%
	Algorithm	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 132: CN284 TDTE - Slice 2 - Quantification Settings for Lobule 2.

		1	2	3	4
Threshold	Min	0	0	0	0
	Max	1092	723	755	755
		96.14%	94.17%	95.07%	92.36%
	Algorithm	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 133: CN284 TDTE - Slice 2 - Quantification Settings for Lobule 3.

		1	2	3	4
Threshold	Min	0	0	0	0
	Max	947	1461	1060	1124
		94.14%	97.01%	94.45%	95.73%
	Algorithm	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 134: CN284 TDTE - Slice 2 - Quantification Settings for Lobule 4.

		1
Threshold	Min	0
	Max	434
		85.19%
	Algorithm	Default
Analyze	Size (micron^2)	17-Inifinity
	Circularity	0.05-1.00

Table 135: CN284 TDTE - Slice 2 - Quantification Settings for Lobule 5.

		1	2	3	4	5	6	7	8	9	10
Threshold	Min	0	0	0	0	0	0	0	0	0	0
	Max	1012	1269	1590	1558	1574	1542	1702	1574	1349	1317
		97.01%	96.76%	96.20%	97.30%	97.20%	97.80%	98.00%	97.40%	97.45%	97.90%
	Algorithm	Default	Default	Default	Default	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 136: CN284 TDTE - Slice 2 - Quantification Settings for Lobule 6.

		1	2	3	4	5
Threshold	Min	0	0	0	0	0
	Max	1397	1783	1461	1108	1028
		97.60%	98.10%	96.45%	96.51%	98.85%
	Algorithm	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 137: CN284 TDTE - Slice 2 - Quantification Settings for Lobule 7.

		1	2	3	4	5
Threshold	Min	0	0	0	0	0
	Max	1397	1108	1718	1510	1686
		97.00%	96.35%	97.15%	97.80%	97.46%
	Algorithm	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 138: CN284 TDTE - Slice 2 - Quantification Settings for Lobule 8.

		1	2	3	4
Threshold	Min	0	0	0	0
	Max	2473	1237	1044	915
		97.15%	95.06%	95.29%	95.28%
	Algorithm	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 139: CN284 TDTE - Slice 2 - Quantification Settings for Lobule 9.

		1	2	3	4	5	6	7
Threshold	Min	0	0	0	0	0	0	0
	Max	1718	1493	915	771	996	2746	1333
		97.06%	98.20%	97.90%	98.94%	96.05%	98.55%	96.02%
	Algorithm	Default	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 140: CN284 TDTE - Slice 2 - Quantification Settings for Lobule 10.

		1	2
Threshold	Min	0	0
	Max	787	674
		95.27%	94.17%
	Algorithm	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00

Table 141: CN284 TDTE - Slice 3 - Quantification Settings for DCN.

		1	2	3
Threshold	Min	0	0	0
	Max	369	401	369
		96.55%	96.25%	96.05%
	Algorithm	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00

Table 142: CN284 TDTE - Slice 3 - Quantification Settings for Lobule 2.

		1	2	3	4
Threshold	Min	0	0	0	0
	Max	610	739	434	626
		96.75%	96.70%	95.97%	95.12%
	Algorithm	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 143: CN284 TDTE - Slice 3 - Quantification Settings for Lobule 3.

		1	2	3	4	5	6
Threshold	Min	0	0	0	0	0	0
	Max	883	691	931	964	996	996
		98.07%	97.15%	97.85%	97.00%	98.00%	97.86%
	Algorithm	Default	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 144: CN284 TDTE - Slice 3 - Quantification Settings for Lobule 4.

		1	2	3	4	5
Threshold	Min	0	0	0	0	0
	Max	739	851	514	723	755
		99.04%	98.00%	99.15%	97.95%	97.40%
	Algorithm	Default	Default	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00	0.05-1.00

Table 145: CN284 TDTE - Slice 3 - Quantification Settings for Lobule 5.

		1	2	3
Threshold	Min	0	0	0
	Max	434	626	642
		98.04%	97.85%	95.71%
	Algorithm	Default	Default	Default
Analyze	Size (micron^2)	17-Inifinity	17-Inifinity	17-Inifinity
	Circularity	0.05-1.00	0.05-1.00	0.05-1.00

Table 146: CN284 TDTE - Slice 3 - Quantification Settings for Lobule 6.

E

CN276 2FD Quantification Results

This appendix displays the quantification results fetched from microglia cell counts with ImageJ regarding a more classical approach. These results refer to each brain image of the animal CN276 2FD.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	14	324.955	23.211	0.485
2	22	554.358	25.198	0.827
3	23	566.255	24.620	0.845
4	15	334.622	22.308	0.499
5	10	281.083	28.108	0.420
6	18	402.662	22.370	0.601
7	21	614.590	29.266	0.917
8	18	447.650	24.869	0.668

Table 147: CN276 2FD - Slice 1 - Automatic Quantification Results for DCN.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	19	429.432	22.602	0.641
2	9	175.491	19.499	0.262
3	11	241.300	21.936	0.360
4	13	278.852	21.450	0.416

Table 148: CN276 2FD - Slice 1 - Automatic Quantification Results for Lobule 7.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	11	314.173	28.561	0.469
2	17	409.355	24.080	0.611
3	9	205.979	22.887	0.307
4	9	194.825	21.647	0.291
5	24	546.178	22.757	0.815
6	15	384.816	25.654	0.574
7	17	464.010	27.295	0.693

Table 149: CN276 2FD - Slice 1 - Automatic Quantification Results for Lobule 8.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	17	470.702	27.688	0.703
2	14	411.585	29.399	0.614
3	28	674.450	24.087	1.007
4	21	495.985	23.618	0.740
5	22	575.178	26.144	0.859
6	18	391.136	21.730	0.584
7	20	498.215	24.911	0.744
8	19	455.086	23.952	0.679
9	22	546.178	24.826	0.815

Table 150: CN276 2FD - Slice 1 - Automatic Quantification Results for Lobule 9.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	10	250.967	25.097	0.375
2	10	228.287	22.829	0.341
3	20	457.317	22.866	0.683
4	10	246.133	24.613	0.367
5	18	452.484	25.138	0.675
6	14	360.648	25.761	0.538
7	5	117.118	23.424	0.174
8	5	147.606	29.521	0.220
9	15	315.289	21.019	0.469

Table 151: CN276 2FD - Slice 1 - Automatic Quantification Results for Lobule 10.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	22	620.167	28.189	0.926
2	12	262.864	21.905	0.392
3	14	342.802	24.486	0.512
4	24	689.322	28.722	1.029
5	27	694.899	25.737	1.037

Table 152: CN276 2FD - Slice 1 - Automatic Quantification Results for Lobule 11.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	16	381.469	23.842	0.934
2	12	291.121	24.260	0.713
3	20	491.523	24.576	1.203
4	13	329.417	25.340	0.807
5	24	651.026	27.126	1.594
6	28	699.360	24.977	1.712
7	15	431.291	28.753	1.056
8	31	780.413	25.175	1.911

Table 153: CN276 2FD - Slice 2 - Automatic Quantification Results for DCN.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	10	282.198	28.220	0.691
2	5	89.976	17.995	0.220
3	5	102.617	20.523	0.251
4	12	256.172	21.348	0.627
5	16	425.342	26.584	1.041
6	6	139.798	23.300	0.342
7	10	214.530	21.453	0.525
8	2	51.309	25.654	0.126

Table 154: CN276 2FD - Slice 2 - Automatic Quantification Results for Lobule 2.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	11	239.813	21.801	0.587
2	19	421.996	22.210	1.033
3	2	56.886	28.443	0.139
4	10	236.466	23.647	0.579
5	14	336.853	24.061	0.825

Table 155: CN276 2FD - Slice 2 - Automatic Quantification Results for Lobule 3.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	13	257.659	19.820	0.631
2	7	170.657	24.380	0.418
3	6	119.720	19.953	0.293
4	8	198.543	24.818	0.486
5	9	216.761	24.085	0.531
6	3	64.322	21.441	0.157

Table 156: CN276 2FD - Slice 2 - Automatic Quantification Results for Lobule 4.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	11	250.967	22.815	0.615
2	15	461.407	30.760	1.131
3	16	435.381	27.211	1.067

Table 157: CN276 2FD - Slice 2 - Automatic Quantification Results for Lobule 5.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	13	330.161	25.397	0.809
2	5	118.605	23.721	0.291
3	14	300.045	21.432	0.735
4	14	356.930	25.495	0.875
5	4	96.297	24.074	0.236
6	7	158.016	22.574	0.387

Table 158: CN276 2FD - Slice 2 - Automatic Quantification Results for Lobule 6.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	10	275.877	27.588	0.676
2	14	316.404	22.600	0.776
3	11	275.134	25.012	0.674
4	10	239.813	23.981	0.588
5	11	255.800	23.255	0.627

Table 159: CN276 2FD - Slice 2 - Automatic Quantification Results for Lobule 7.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	11	271.416	24.674	0.665
2	9	205.979	22.887	0.505
3	7	212.671	30.382	0.521
4	7	184.042	26.292	0.451
5	5	118.233	23.647	0.290
6	7	181.068	25.867	0.444
7	12	308.224	25.685	0.756

Table 160: CN276 2FD - Slice 2 - Automatic Quantification Results for Lobule 8.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	15	317.519	21.168	0.777
2	8	198.171	24.771	0.485
3	12	292.237	24.353	0.716
4	5	122.323	24.465	0.300
5	12	272.903	22.742	0.668
6	11	273.647	24.877	0.670
7	13	307.481	23.652	0.755

Table 161: CN276 2FD - Slice 2 - Automatic Quantification Results for Lobule 9.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	6	127.156	21.193	0.312
2	11	251.338	22.849	0.617
3	9	188.504	20.945	0.463
4	2	55.027	27.513	0.134
5	9	191.478	21.275	0.467
6	13	298.929	22.995	0.731
7	13	306.365	23.567	0.749

Table 162: CN276 2FD - Slice 2 - Automatic Quantification Results for Lobule 10.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	4	87.745	21.936	0.215
2	7	174.375	24.911	0.428
3	4	40.798	10.199	0.100
4	3	66.924	22.308	0.164
5	8	232.377	29.047	0.570
6	15	395.970	26.398	0.972
7	17	343.174	20.187	0.842
8	8	217.876	27.235	0.535
9	18	425.342	23.630	1.0044

Table 163: CN276 2FD - Slice 2 - Automatic Quantification Results for Lobule 11.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	15	389.277	25.952	0.952
2	13	305.993	23.538	0.755
3	12	266.954	22.246	0.658
4	6	142.400	23.733	0.349
5	4	92.207	23.502	0.226
6	11	281.454	25.587	0.689

Table 164: CN276 2FD - Slice 2 - Automatic Quantification Results for Lobule 12.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	18	490.036	27.224	1.200
2	10	219.363	21.936	0.537
3	6	154.670	25.778	0.379
4	8	172.145	21.518	0.421
5	11	278.108	25.283	0.681

Table 165: CN276 2FD - Slice 2 - Automatic Quantification Results for Lobule 13.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	6	161.734	26.956	0.396
2	11	286.660	26.060	0.702
3	2	44.244	22.122	0.108
4	2	62.463	31.231	0.153
5	11	279.224	25.384	0.684
6	11	242.043	22.004	0.593
7	9	191.850	21.317	0.470

Table 166: CN276 2FD - Slice 2 - Automatic Quantification Results for Lobule 14.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	12	239.069	19.922	0.585
2	7	185.529	26.504	0.454
3	3	58.373	19.458	0.143
4	6	123.810	20.635	0.303
5	5	115.631	23.126	0.283

Table 167: CN276 2FD - Slice 2 - Automatic Quantification Results for Lobule 15.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	7	140.913	20.130	0.345
2	6	166.196	27.699	0.407
3	10	264.352	26.435	0.647
4	6	120.836	20.139	0.296

Table 168: CN276 2FD - Slice 2 - Automatic Quantification Results for Lobule 16.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	8	206.350	25.794	0.505
2	4	123.810	30.953	0.303
3	2	42.757	21.379	0.105
4	13	313.801	24.139	0.768
5	11	278.480	25.316	0.682
6	7	165.824	23.689	0.406
7	7	173.632	24.805	0.425
8	12	286.288	23.857	0.701

Table 169: CN276 2FD - Slice 2 - Automatic Quantification Results for Lobule 17.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	7	153.554	21.936	0.376
2	6	147.234	24.539	0.360
3	12	298.186	24.849	0.730
4	4	118.233	29.556	0.290
5	3	70.642	23.547	0.173
6	7	164.337	23.477	0.403
7	4	73.989	18.497	0.182
8	12	278.852	23.238	0.684
9	12	295.955	24.663	0.726
10	14	327.186	23.370	0.803

Table 170: CN276 2FD - Slice 2 - Automatic Quantification Results for Lobule 18.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	19	515.690	27.142	1.344
2	19	413.444	21.760	1.077
3	23	605.294	26.317	1.577
4	18	464.381	25.799	1.210
5	29	811.273	27.975	2.114
6	15	426.829	28.455	1.112
7	30	759.221	25.307	1.978
8	39	971.148	24.901	2.530

Table 171: CN276 2FD - Slice 3 - Automatic Quantification Results for DCN.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	47	1302.052	27.703	3.393
2	43	1177.127	27.375	3.067

Table 172: CN276 2FD - Slice 3 - Automatic Quantification Results for Lobule 2.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	10	204.120	20.412	0.532
2	8	217.504	27.188	0.567
3	18	478.882	26.605	1.248
4	10	281.454	28.145	0.733
5	10	222.338	22.234	0.579
6	14	337.968	24.141	0.881
7	14	314.173	22.441	0.819
8	5	121.579	24.316	0.317
9	16	365.854	22.866	0.953
10	11	276.621	25.147	0.721

Table 173: CN276 2FD - Slice 3 - Automatic Quantification Results for Lobule 3.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	8	203.004	25.376	0.529
2	2	48.334	24.167	0.126
3	4	84.399	21.100	0.220
4	7	156.157	22.308	0.407
5	5	115.631	23.126	0.301
6	4	79.938	19.984	0.208
7	5	156.901	31.380	0.409
8	6	158.388	26.398	0.413
9	5	112.656	22.531	0.294
10	2	39.411	19.706	0.103
11	6	153.183	25.530	0.399
12	3	63.206	21.069	0.165

Table 174: CN276 2FD - Slice 3 - Automatic Quantification Results for Lobule 4.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	8	210.812	26.352	0.549
2	5	108.195	21.639	0.282
3	6	129.015	21.503	0.336
4	7	153.926	21.989	0.401
5	4	89.604	22.401	0.233
6	7	169.914	24.273	0.443

Table 175: CN276 2FD - Slice 3 - Automatic Quantification Results for Lobule 5.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	9	223.825	24.869	0.583
2	14	348.751	24.911	0.909
3	18	505.651	28.092	1.318
4	7	171.773	24.539	0.448
5	13	388.905	29.916	1.013

Table 176: CN276 2FD - Slice 3 - Automatic Quantification Results for Lobule 6.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	6	171.401	28.567	0.447
2	7	164.709	23.530	0.429
3	15	432.778	28.852	1.128
4	9	236.838	26.315	0.617
5	15	354.328	23.622	0.923
6	12	278.852	23.238	0.727
7	16	361.020	22.564	0.941
8	9	205.235	22.804	0.535
9	12	304.506	25.376	0.793

Table 177: CN276 2FD - Slice 3 - Automatic Quantification Results for Lobule 7.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	5	107.079	21.416	0.279
2	3	81.053	27.018	0.211
3	12	272.903	22.742	0.711
4	6	128.644	21.441	0.335
5	6	120.836	20.139	0.315
6	7	166.939	23.848	0.435
7	5	116.002	23.200	0.302
8	5	166.568	33.314	0.434
9	3	86.258	28.753	0.225
10	9	224.569	24.952	0.585
11	7	160.247	22.892	0.418
12	6	168.798	28.133	0.440
13	5	123.438	24.688	0.322
14	8	236.466	29.558	0.616

Table 178: CN276 2FD - Slice 3 - Automatic Quantification Results for Lobule 8.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	7	186.645	26.664	0.486
2	8	181.440	22.680	0.473
3	9	222.338	24.704	0.579
4	11	262.121	23.829	0.683
5	13	299.673	23.052	0.781
6	7	142.029	20.290	0.370
7	8	175.491	21.936	0.457
8	11	295.211	26.837	0.769
9	14	363.251	25.947	0.946

Table 179: CN276 2FD - Slice 3 - Automatic Quantification Results for Lobule 9.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	10	269.557	26.956	0.704
2	7	191.478	27.354	0.500
3	10	205.979	20.598	0.538
4	3	65.809	21.936	0.172
5	6	160.619	26.770	0.420
6	7	164.709	23.530	0.430
7	11	265.095	24.100	0.693
8	14	292.237	20.874	0.763
9	12	249.479	20.790	0.652
10	7	149.836	21.405	0.391

Table 180: CN276 2FD - Slice 3 - Automatic Quantification Results for Lobule 10.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	8	275.877	34.485	0.721
2	12	290.006	24.167	0.758
3	4	76.963	19.241	0.201
4	6	134.593	22.432	0.352
5	4	78.079	19.520	0.204
6	5	136.452	27.290	0.356
7	9	197.427	21.936	0.516
8	13	279.224	21.479	0.729

Table 181: CN276 2FD - Slice 3 - Automatic Quantification Results for Lobule 11.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	5	121.208	24.242	0.316
2	6	138.682	23.114	0.361
3	5	131.618	26.324	0.343
4	7	192.222	27.460	0.501
5	8	165.452	20.682	0.431
6	5	101.130	20.226	0.264
7	10	224.197	22.420	0.586

Table 182: CN276 2FD - Slice 3 - Automatic Quantification Results for Lobule 12.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	6	179.209	29.868	0.467
2	4	122.323	30.581	0.319
3	9	243.159	27.018	0.634
4	12	292.980	24.415	0.736

Table 183: CN276 2FD - Slice 3 - Automatic Quantification Results for Lobule 13.

F

CN282 2TE Quantification Results

This appendix displays the quantification results fetched from microglia cell counts with ImageJ regarding a more classical approach. These results refer to each brain image of the animal CN282 2TE.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	37	977.064	26.407	0.715
2	28	878.345	31.369	0.643
3	32	884.673	27.646	0.647
4	35	1012.501	28.929	0.741
5	21	668.251	31.821	0.490
6	26	797.977	30.691	0.585

Table 184: CN282 2TE - Slice 1 - Automatic Quantification Results for DCN.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	20	592.946	29.647	0.435
2	15	377.157	25.144	0.277
3	27	785.954	29.109	0.576
4	31	899.860	29.028	0.660
5	25	642.938	25.718	0.472
6	33	936.564	28.381	0.687
7	24	536.626	22.359	0.394
8	34	1013.767	29.817	0.744

Table 185: CN282 2TE - Slice 1 - Automatic Quantification Results for Lobule 2.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	25	706.852	28.274	0.518
2	27	696.727	25.805	0.511
3	35	923.907	26.397	0.678
4	26	791.649	30.448	0.581
5	30	696.727	23.224	0.511

Table 186: CN282 2TE - Slice 1 - Automatic Quantification Results for Lobule 3.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	22	649.899	29.541	0.477
2	28	699.891	24.996	0.513
3	20	513.212	25.661	0.376
4	21	544.219	25.915	0.399
5	26	679.641	26.140	0.498
6	23	627.118	27.266	0.460
7	21	547383	26.066	0.401
8	20	472.711	23.636	0.347

Table 187: CN282 2TE - Slice 1 - Automatic Quantification Results for Lobule 4.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	20	683.438	34.172	0.501
2	26	818.860	31.495	0.603
3	18	473.344	26.297	0.348
4	21	658.759	31.369	0.485
5	24	774.563	32.273	0.570
6	16	509.415	31.838	0.375
7	28	720.774	25.742	0.533
8	20	539.790	26.989	0.399

Table 188: CN282 2TE - Slice 1 - Automatic Quantification Results for Lobule 5.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	29	915.681	31.575	0.674
2	22	612.563	27.844	0.451
3	23	637.876	27.734	0.469
4	18	564.469	31.359	0.415
5	32	894.798	27.962	0.659
6	36	1041.611	28.934	0.767

Table 189: CN282 2TE - Slice 1 - Automatic Quantification Results for Lobule 6.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	50	1279.548	25.591	0.445
2	63	1576.338	25.021	0.548
3	26	648.634	24.947	0.671

Table 190: CN282 2TE - Slice 1 - Automatic Quantification Results for Lobule 7.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	17	475.876	27.993	0.350
2	20	485.368	24.268	0.357
3	20	493.594	24.680	0.363

Table 191: CN282 2TE - Slice 1 - Automatic Quantification Results for Lobule 8.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	40	1066.923	26.673	0.956
2	35	965.673	27.591	0.865
3	27	802.407	29.719	0.719
4	16	471.446	29.465	0.422
5	29	890.368	30.702	0.798
6	22	728.368	33.108	0.653
7	26	1006.806	38.723	0.902

Table 192: CN282 2TE - Slice 2 - Automatic Quantification Results for DCN.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	15	339.188	22.613	0.305
2	10	257.555	25.755	0.231
3	16	444.868	27.804	0.399
4	24	822.657	34.277	0.739
5	19	485.368	25.546	0.436
6	21	568.899	27.090	0.511
7	10	246.164	24.616	0.221
8	12	283.500	23.625	0.255
9	18	450.563	25.031	0.405

Table 193: CN282 2TE - Slice 2 - Automatic Quantification Results for Lobule 2.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	22	596.110	27.096	0.535
2	11	318.305	28.937	0.286
3	26	764.438	29.401	0.686
4	27	742.290	27.492	0.666
5	11	254.391	23.126	0.228
6	13	327.164	25.166	0.294
7	31	931.501	30.048	0.836
8	26	708.118	27.235	0.636

Table 194: CN282 2TE - Slice 2 - Automatic Quantification Results for Lobule 3.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	20	551.180	27.559	0.495
2	18	475.243	26.402	0.427
3	23	623.321	27.101	0.560
4	15	346.149	23.077	0.311
5	24	628.384	26.183	0.564
6	22	663.188	30.145	0.595
7	10	361.336	36.134	0.324
8	8	279.703	34.963	0.252
9	16	492.962	30.810	0.445

Table 195: CN282 2TE - Slice 2 - Automatic Quantification Results for Lobule 4.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	10	327.797	32.780	0.295
2	20	480.939	24.047	0.433
3	14	392.977	28.070	0.354
4	11	318.938	28.994	0.287
5	23	585.352	25.450	0.527
6	11	342.985	31.180	0.309
7	22	685.970	31.180	0.618
8	12	382.852	31.904	0.345
9	4	122.133	30.533	0.110

Table 196: CN282 2TE - Slice 2 - Automatic Quantification Results for Lobule 5.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	10	254.391	25.439	0.229
2	16	467.016	29.189	0.418
3	16	448.032	28.002	0.403
4	12	315.141	26.262	0.284
5	14	360.071	25.719	0.324
6	28	657.493	23.482	0.592
7	14	382.852	27.347	0.345
8	16	565.735	35.358	0.509
9	16	499.922	31.245	0.450

Table 197: CN282 2TE - Slice 2 - Automatic Quantification Results for Lobule 6.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	17	436.641	25.685	0.391
2	4	78.496	19.617	0.391
3	10	256.289	25.629	0.230
4	19	410.063	21.582	0.368
5	18	425.250	23.625	0.382

Table 198: CN282 2TE - Slice 2 - Automatic Quantification Results for Lobule 7.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	32	854.931	26.717	0.816
2	34	851.134	25.033	0.812
3	24	657.493	27.396	0.627
4	39	1225.126	31.413	1.169
5	19	586.618	30.875	0.560
6	40	1593.424	39.836	1.520
7	24	796.079	33.170	0.759

Table 199: CN282 2TE - Slice 4 - Automatic Quantification Results for DCN.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	14	311.977	22.284	0.298
2	16	419.555	26.222	0.400
3	31	830.251	26.782	0.792
4	31	946.056	30.518	0.903
5	15	353.743	23.583	0.337
6	20	520.805	26.040	0.497
7	45	1239.048	27.534	1.182
8	15	336.657	22.444	0.321
9	24	672.680	28.028	0.642

Table 200: CN282 2TE - Slice 4 - Automatic Quantification Results for Lobule 2.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	24	614.462	25.603	0.586
2	22	572.696	26.032	0.546
3	23	661.290	28.752	0.631
4	16	403.102	25.194	0.385
5	11	274.008	24.910	0.261
6	22	591.680	26895	0.564
7	22	644.204	29.282	0.615

Table 201: CN282 2TE - Slice 4 - Automatic Quantification Results for Lobule 3.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	20	577.126	28.856	0.551
2	11	363.868	33.079	0.347
3	15	384.750	25.650	0.367
4	21	530.297	25.252	0.506
5	16	353.110	22.069	0.337
6	10	251.227	25.123	0.240
7	8	235.407	29.426	0.225
8	11	249.328	22.666	0.238
9	10	270.211	27.021	0.258
10	12	282.993	24.416	0.280
11	15	346.149	23.077	0.330

Table 202: CN282 2TE - Slice 4 - Automatic Quantification Results for Lobule 4.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	17	514.477	30.263	0.491
2	11	313.243	28.47	0.299
3	12	365.766	30.481	0.349
4	15	367.664	24.511	0.351
5	4	149.977	37494	0.143
6	6	122.766	20.461	0.117
7	11	30.914	27.901	0.293
8	18	487.899	27.106	0.465
9	11	299.953	27.268	0.286
10	6	145.547	24.258	0.139

Table 203: CN282 2TE - Slice 4 - Automatic Quantification Results for Lobule 5.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	17	563.837	33.167	0.538
2	19	544.852	28.676	0.520
3	27	776.462	28.758	0.741
4	22	593.579	26.981	0.566
5	21	575.860	27.422	0.549
6	35	1045.407	29.869	0.997

Table 204: CN282 2TE - Slice 4 - Automatic Quantification Results for Lobule 6.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	12	315.141	26.262	0.301
2	18	440.438	24.469	0.422
3	14	337.922	24.137	0.324
4	14	394.875	28.205	0.379
5	15	405.000	27.000	0.388
6	17	425.250	25.015	0.408
7	11	299.321	27.211	0.287

Table 205: CN282 2TE - Slice 4 - Automatic Quantification Results for Lobule 7.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	15	377.789	25.186	0.362
2	18	512.579	28.477	0.491
3	21	525.235	25.011	0.504

Table 206: CN282 2TE - Slice 4 - Automatic Quantification Results for Lobule 8.

G

CN283 2FD Quantification Results

This appendix displays the quantification results fetched from microglia cell counts with ImageJ regarding a more classical approach. These results refer to each brain image of the animal CN283 2FD.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	35	1026.423	29.326	0.644
2	25	806.204	32.248	0.506
3	23	724.751	31.503	0.454
4	30	873.282	29.109	0.548
5	32	835.313	26.104	0.524
6	17	754.313	44.371	0.473

Table 207: CN283 2FD - Slice 1 - Automatic Quantification Results for DCN.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	14	379.688	27.121	0.238
2	13	323.368	24.874	0.203
3	9	204.399	22.711	0.128
4	21	566.363	26.970	0.355
5	17	747.985	43.999	0.469
6	9	224.016	24.891	0.141
7	6	136.688	22.781	0.086
8	7	179.719	25.674	0.113
9	5	106.945	21.389	0.067
10	16	405.633	25.352	0.254

Table 208: CN283 2FD - Slice 1 - Automatic Quantification Results for Lobule 2.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	25	640.407	25.616	0.402
2	34	953.650	28.049	0.598
3	24	748.618	31.192	0.470
4	27	720.141	26.672	0.452
5	5	153.774	30.755	0.096
6	8	199.969	24996	0.125
7	20	562.571	28.129	0.353
8	15	399.305	26.620	0.250

Table 209: CN283 2FD - Slice 1 - Automatic Quantification Results for Lobule 3.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	31	833.415	26.884	0.523
2	13	346.149	26.627	0.217
3	9	226.547	25.172	0.142
4	30	776.462	25.882	0.487
5	9	202.500	22.500	0.127
6	12	309.446	25.787	0.194
7	18	520.805	28.934	0.327

Table 210: CN283 2FD - Slice 1 - Automatic Quantification Results for Lobule 4.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	33	942.892	28.572	0.591
2	36	1070.087	29.725	0.671
3	14	341.719	24.409	0.214
4	11	268.313	24.392	0.168
5	14	425.250	30.375	0.267
6	17	465.751	27.397	0.292
7	4	106.945	26.736	0.067

Table 211: CN283 2FD - Slice 1 - Automatic Quantification Results for Lobule 5.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	19	560.673	29.509	0.352
2	15	368.297	24.553	0.231
3	12	385.383	32.115	0.242
4	15	398.672	26.578	0.250
5	25	687.868	27.515	0.431
6	19	480.938	25.313	0.302
7	6	153.141	25.523	0.096

Table 212: CN283 2FD - Slice 1 - Automatic Quantification Results for Lobule 6.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	11	538.524	48.957	0.338
2	22	753.048	34.229	0.472
3	13	311.344	23.950	0.195
4	14	440.438	31.460	0.276
5	12	306.071	30.006	0.226
6	21	604.337	28.778	0.379
7	19	467.649	24.613	0.293
8	23	906.189	39.400	0.568

Table 213: CN283 2FD - Slice 1 - Automatic Quantification Results for Lobule 7.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	39	1151.720	29.531	0.631
2	35	944.790	26.994	0.518
3	31	832.149	26.844	0.456
4	21	563.837	26.849	0.309
5	18	489.165	27.176	0.268
6	28	811.267	28.974	0.445

Table 214: CN283 2FD - Slice 4 - Automatic Quantification Results for DCN.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	17	451.829	26.578	0.248
2	25	618.259	24.730	0.339
3	24	523.337	21.806	0.287
4	15	370.829	24.722	0.203
5	26	713.813	27.454	0.391

Table 215: CN283 2FD - Slice 4 - Automatic Quantification Results for Lobule 2.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	30	861.892	28.730	0.472
2	27	695.462	25.758	0.381
3	42	1084.009	25.810	0.594
4	24	628.384	26.183	0.344

Table 216: CN283 2FD - Slice 4 - Automatic Quantification Results for Lobule 3.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	29	825.188	28.455	0.452
2	39	1162.478	29.807	0.637
3	31	787.852	25.415	0.432
4	24	620.157	25.840	0.340
5	23	591.048	25.698	0.324
6	14	384.118	27.437	0.211

Table 217: CN283 2FD - Slice 4 - Automatic Quantification Results for Lobule 4.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	19	578.391	30.442	0.317
2	26	750.516	28.866	0.411
3	7	278.438	39.777	0.153
4	26	680.274	26.164	0.373
5	48	1468.759	30.599	0.805
6	33	809.368	24.526	0.444

Table 218: CN283 2FD - Slice 4 - Automatic Quantification Results for Lobule 5.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	29	831.517	28.673	0.456
2	25	708.751	28.350	0.388
3	7	162.000	23.143	0.089
4	22	602.438	27.384	0.330
5	28	959.978	34.285	0.526
6	17	548.649	32.273	0.301

Table 219: CN283 2FD - Slice 4 - Automatic Quantification Results for Lobule 6.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	26	680.907	26.189	0.373
2	19	587.251	30.908	0.322
3	28	747.985	26.714	0.410
4	11	305.649	27.786	0.168

Table 220: CN283 2FD - Slice 4 - Automatic Quantification Results for Lobule 7.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	18	591.048	32.836	0.324
2	13	298.688	22.976	0.164
3	28	944.157	33.720	0.517

Table 221: CN283 2FD - Slice 4 - Automatic Quantification Results for Lobule 8.

H

CN284 TDTE Quantification Results

This appendix displays the quantification results fetched from microglia cell counts with ImageJ regarding a more classical approach. These results refer to each brain image of the animal CN284 TDTE.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	26	794.170	30.545	0.953
2	21	523.870	24.946	0.629
3	13	300.788	23.138	0.361
4	27	712.002	26.370	0.854
5	33	868.902	36.330	1.043
6	16	416.047	26.003	0.499
7	23	512.716	22.292	0.615

Table 222: CN284 TDTE - Slice 1 - Automatic Quantification Results for DCN.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	15	365.854	24.390	0.439
2	14	316.776	22.627	0.380
3	14	369.943	26.425	0.444
4	16	369.943	23.121	0.444
5	13	343.546	26.427	0.412
6	19	504.536	26.555	0.605
7	24	655.488	27.312	0.787

Table 223: CN284 TDTE - Slice 1 - Automatic Quantification Results for Lobule 2.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	10	290.006	29.001	0.348
2	11	239.813	21.801	0.288
3	23	486.318	21.144	0.584
4	5	155.042	31.008	0.186
5	11	299.301	27.209	0.359
6	15	353.584	23.572	0.424

Table 224: CN284 TDTE - Slice 1 - Automatic Quantification Results for Lobule 3.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	21	473.676	22.556	0.568
2	14	302.275	21.591	0.363
3	20	515.318	25.766	0.618
4	13	285.916	21.994	0.343
5	35	858.120	24.518	1.030
6	6	182.183	30.364	0.219
7	20	455.086	22.754	0.546

Table 225: CN284 TDTE - Slice 1 - Automatic Quantification Results for Lobule 4.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	7	171.773	24.539	0.206
2	13	344.289	26.484	0.413
3	9	172.516	19.168	0.207
4	12	252.454	21.038	0.303
5	7	146.490	20.927	0.176
6	9	211.184	23.465	0.256

Table 226: CN284 TDTE - Slice 1 - Automatic Quantification Results for Lobule 5.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	15	292.609	19.507	0.351
2	19	454.343	23.913	0.545
3	24	511.600	21.317	0.614
4	26	675.937	25.998	0.811
5	28	595.628	21.272	0.715
6	6	126.785	21.131	0.152

Table 227: CN284 TDTE - Slice 1 - Automatic Quantification Results for Lobule 6.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	13	360.648	27.742	0.433
2	21	536.883	25.566	0.644
3	16	571.832	35.740	0.686
4	7	146.490	20.927	0.176
5	17	423.855	24.933	0.509
6	17	407.124	23.948	0.489
7	18	397.829	22.102	0.477
8	13	274.390	21.107	0.329

Table 228: CN284 TDTE - Slice 1 - Automatic Quantification Results for Lobule 7.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	23	556.588	24.199	0.608
2	25	622.397	24.896	0.680
3	65	1895.821	29.166	2.072
4	67	1841.538	27.486	2.013
5	32	820.940	25.654	0.897

Table 229: CN284 TDTE - Slice 2 - Automatic Quantification Results for DCN.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	20	521.267	26.063	0.570
2	26	683.745	26.298	0.747
3	27	671.475	24.869	0.734
4	29	727.246	25.077	0.795

Table 230: CN284 TDTE - Slice 2 - Automatic Quantification Results for Lobule 2.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	14	304.134	21.724	0.332
2	26	611.987	23.538	0.669
3	31	844.363	27.238	0.923
4	25	697.501	27.900	0.762

Table 231: CN284 TDTE - Slice 2 - Automatic Quantification Results for Lobule 3.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	26	709.771	27.299	0.776
2	10	367.341	36.734	0.401
3	27	809.042	29.965	0.884
4	26	745.464	28.672	0.815

Table 232: CN284 TDTE - Slice 2 - Automatic Quantification Results for Lobule 4.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	165	4442.296	26.923	4.855

Table 233: CN284 TDTE - Slice 2 - Automatic Quantification Results for Lobule 5.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	12	292.980	24.415	0.320
2	8	187.017	23.377	0.204
3	14	285.544	20.396	0.312
4	6	133.105	22.184	0.145
5	13	377.751	29.058	0.413
6	7	168.055	24.008	0.184
7	4	92.951	23.238	0.102
8	8	207.466	25.933	0.227
9	5	154.670	30.934	0.169
10	8	184.786	23.098	0.202

Table 234: CN284 TDTE - Slice 2 - Automatic Quantification Results for Lobule 6.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	8	207.466	25.933	0.227
2	5	97.412	19.482	0.106
3	9	237.210	26.357	0.259
4	15	399.688	26.646	0.437
5	19	432.035	22.739	0.472

Table 235: CN284 TDTE - Slice 2 - Automatic Quantification Results for Lobule 7.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	12	292.237	24.353	0.319
2	21	560.678	26.699	0.613
3	10	709.027	70.903	0.775
4	4	126.785	31.696	0.139
5	16	504.536	31.533	0.551

Table 236: CN284 TDTE - Slice 2 - Automatic Quantification Results for Lobule 8.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	29	750.297	25.872	0.820
2	17	408.983	24.058	0.447
3	24	581.871	24.245	0.636
4	23	533.908	23.213	0.583

Table 237: CN284 TDTE - Slice 2 - Automatic Quantification Results for Lobule 9.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	14	392.995	28.071	0.429
2	5	122.323	24.465	0.134
3	8	167.683	20.960	0.183
4	3	97.784	32.595	0.107
5	28	888.608	31.736	0.971
6	12	274.018	22.835	0.299
7	18	442.817	24.601	0.484

Table 238: CN284 TDTE - Slice 2 - Automatic Quantification Results for Lobule 10.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	60	1481.633	24.694	0.838
2	59	2020.003	34.237	1.143

Table 239: CN284 TDTE - Slice 3 - Automatic Quantification Results for DCN.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	22	548.780	24.945	0.311
2	31	759.593	24.503	0.430
3	36	868.159	24.116	0.491

Table 240: CN284 TDTE - Slice 3 - Automatic Quantification Results for Lobule 2.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	29	818.337	28.219	0.463
2	36	844.735	23.465	0.478
3	42	1042.906	24.831	0.590
4	37	949.955	25.674	0.538

Table 241: CN284 TDTE - Slice 3 - Automatic Quantification Results for Lobule 3.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	10	249.108	24.911	0.141
2	16	397.085	24.818	0.225
3	12	312.686	26.057	0.177
4	12	253.198	21.100	0.143
5	10	26.138	26.138	0.148
6	14	321.981	22.999	0.182

Table 242: CN284 TDTE - Slice 3 - Automatic Quantification Results for Lobule 4.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	4	101.130	25.283	0.057
2	16	362.507	22.657	0.205
3	6	118.233	19.706	0.067
4	18	456.945	25.386	0.259
5	20	475.907	23.795	0.269

Table 243: CN284 TDTE - Slice 3 - Automatic Quantification Results for Lobule 5.

Image	Number of Cells	Total Area (Pixels)	Average Size (Pixels)	Area (%)
1	12	285.916	23.826	0.162
2	20	467.356	23.368	0.364
3	25	604551	24.182	0.342

Table 244: CN284 TDTE - Slice 3 - Automatic Quantification Results for Lobule 6.