Alma Mater Studiorum · University of Bologna

School of Science
Department of Physics and Astronomy
Master Degree in Physics

Deep learning-based tool for radiomics analysis of cancer 3D multicellular spheroids

Supervisor:
Prof. Gastone Castellani

Co-supervisors:
Prof. Filippo Piccinini
Prof. Giovanni Martinelli

Submitted by:
Mariachiara Stellato

Academic Year 2022/2023

“*Natures uses only the longest threads to weave her patterns,*

so each small piece of her fabric reveals

the organization of the entire tapestry.”

Richard Feynman

Key Words

Microscopy

Spheroids

Convolutional Neural Networks

Image segmentation

Radiomics

Matlab

Abstract

Cancer 3D multicellular spheroids are a fundamental *in vitro* tool for studying *in vivo* tumors. Volume is the main feature used for evaluating the drug and treatment effects, but several other features can be estimated even from a simple 2D image. For high-content screening analysis, the bottleneck is the segmentation stage, which is essential for detecting the spheroids in the images and then proceeding to the feature extraction stage for performing radiomic analysis. Thanks to new deep learning models, it is possible to optimize the process for adapting the analysis to big datasets. One of the most promising approaches is the use of convolutional neural networks (CNNs), which have shown remarkable results in various medical imaging applications. By training a CNN on a large dataset of annotated images, it can learn to recognize patterns and features that are relevant for segmenting spheroids in new images. This approach has several advantages, such as manual or semi-automatic segmentation, which are time-consuming and prone to inter-observer variability. Moreover, CNNs can be fine-tuned for specific tasks and can handle different types of data, such as multi-modal or multi-dimensional images. Starting from the first version of Analysis of SPheroids (*AnaSP)*, an open-source software for estimating morphological features of spheroids, we implemented a new module for automatically segmenting brightfield images by exploiting CNNs. In this work, several deep learning segmentation models have been trained and compared using ground truth masks. Then, a module based on an 18-layer deep residual network (*ResNet18*) was integrated into *AnaSP*, releasing *AnaSP 2.0,* a version of the tool optimized for high-content screening analysis.

Index

Table of figures

Introduction

This thesis project is born within the collaborations of the research group *"BioPhysics"*, formed by professors at the University of Bologna (UniBo) and the unit *"Microscopy & Artificial Intelligence"* (MiAi), which is composed of the researchers of *"IRCCS Istituto Romagnolo dei Tumori Dino Amadori"* (IRST), located in Meldola (FC). This collaboration has been initiated with the purpose of coordinating experts and resources from the technology information and the engineering and physics information fields, both from the biomedical, biological, chemical and medical spheres of study. Specifically, the thesis project "*Deep learning-based tool for radiomics analysis of cancer 3D multicellular spheroids"* is a follow up of a series of previous projects, that led to the creation of the first form of the software termed Analysis of Spheroids (*AnaSP*). The project's aim is to enhance, extend and optimize a tool for the segmentation of spheroids and the following extraction of features for the radiomic analysis.

AnaSP fills out some of the requests expressed by life scientists of IRST. To better understand the operating framework, it is necessary to specify that many of the research groups of IRST utilize tumoral spheroids to test chemotherapeutic drugs and radiotherapy treatments in High-Content Screening (HCS) experiments. However, there is no a validated software for segmenting and extracting features of tumoral spheroids in order to proceed with the radiomic analysis. Due to this necessity, *AnaSP* has been created with the purpose of fill up this lack of technique and automatism regarding the process of validation of medication and anti-tumoral treatments.

The acknowledgement of the problem and the first hypothesis for a solution stemmed from the original work of Prof. Filippo Piccinini. The original tool was tested using tumoral spheroids images, made available thanks to Prof. Giovanni Martinelli, scientific director of IRST. For the final realization of the tool and the associated experiments, it was sought a student mastering at the Department of Physics and the right person was found thanks to Prof. Gastone Castellani, director of the School of Specialization in Medical Physic at the University of Bologna.

The first publication of the first release of *AnaSP* dates back to 2015 in the article "*F. Piccinini, AnaSP: a software suite for automatic image analysis of multicellular spheroids. Computer Methods and Programs in Biomedicine, 119(1):43-52, 2015. DOI: 10.1016/j.cmpb.2015.02.006.*" [1]. This tool was implemented in MATLAB and was available as open-source code and standalone version for Windows operative systems. It was based on the spheroids' segmentation through thresholding-based methods.

In this thesis project, *AnaSP* has been substantially extended. In particular, two new modules have been developed.

(I) The first one enables the automatic segmentation of spheroids through using pre-trained deep learning networks. The trainings were performed by using the images contained in the scientific article: "*A. Peirsman, E. Blondeel, T. Ahmed, J. Anckaert, D.*

Audenaert, T. Boterberg, K. Buzas, N. Carragher, G. Castellani, F. Castro, V. Dangles-Marie, J. Dawson, P. De Tullio, E. De Vlieghere, S. Dedeyne, H. Depypere, A. Diosdi, R.I. Dmitriev, H. Dolznig, S. Fischer, C. Gespach, V. Goossens, J.Heino, A. Hendrix, P. Horvath, L. A. Kunz-Schughart, S. Maes, C. Mangodt, P. Mestdagh, S. Michlíková, M.J. Oliveira, F. Pampaloni, F. Piccinini, C. Pinheiro, J. Rahn, S.M. Robbins, E. Siljamäki, P. Steigemann, G. Sys, S. Takayama, A. Tesei, J. Tulkens, M. Van Waeyenberge, J. Vandesompele, G. Wagemans, C. Weindorfer, N. Yigit, N. Zablowsky, M. Zanoni, P. Blondeel, O. De Wever, MISpheroID: a knowledgebase and transparency tool for minimum information in spheroid identity. Nature Methods, 18:1294-1303, 2021. DOI: 10.1038/s41592-021-01291-4” [2]. 4 trained networks with these images are available to the users;

(II) The second module allows the user to train new deep learning networks by using new images datasets provided by the user himself. In this way the user can obtain specific models created fort the different cases of interest.

Moreover, standalone versions have been developed for Mac and Linux as well as for Windows. The final product is accompanied by a user manual and a video tutorial that better describe each new developed feature.

In the following chapters of this thesis, the work done to develop this tool is explained in detail.

1 Spheroids: 3D in vitro models for drug testing

Today, there are several types of *in vitro* models used in laboratory for testing drugs. Between them, cancer spheroids have a special role because they allow a standardization of the data acquired. In particular, spheroids are multicellular tridimensional (3D) systems with a predominance of cell-cell interactions over cell-substrate interactions. They enable us to mimic *in vitro* some histo-morphological, functional, and environmental characteristics of tumour tissues, essentially serving as tumour models for oncological studies.

They can be derived from different cell types, including tumor cells, primary cells, or induced pluripotent stem cells (iPSCs). Spheroids exhibit several key characteristics that resemble *in vivo* tissues, including cell-cell interaction, spatial organization, extracellular matrix (ECM) deposition, and gradients of oxygen, nutrients and signalling molecules. These features contribute to the recapitulation of tissue complexity and function in spheroid cultures [3].

The use of three-dimensional cell cultures dates to the early 20th century when researchers recognized the limitations of traditional two-dimensional (2D) monolayers cultures in mimicking the complexity of tissues and organs. However, it was not until the 1970s that spheroids, also known as multicellular tumour spheroids (MTS), gained significant attention as an *in vitro* model for studying tumour behaviour and drug responses. Since then, spheroids have evolved as a versatile tool beyond cancer research, finding applications in various fields such as regenerative medicine, neurobiology, developmental biology and drug discovery.

Initially, tumour spheroid cultures were applied in experimental radiotherapy followed by photodynamic treatment, hyperthermia, and chemotherapy including target-specific approaches as well as other contemporary lines of attack such as gene therapy or cell and antibody-based immunotherapy.

They are sandwiched between a monolayer 2D cultures and animal models. Since of its 3D structure, 3D cultures are an excellent experimental model on which to assess the effects of medications and therapeutic treatments because they can reproduce the genuine conditions of tumor cells more accurately. 3D aggregates made of cells with diverse phenotypes are able to better replicate actual tumors characterized by an essentially spherical form. As solid tumors investigated *in vivo*, they are distinguished by a spatial proliferation gradient. This means that tumor cells on the surface of the spheroids are full of oxygen and nutrients, indicating that they are actively proliferating, whereas replication time is longer deeper in the spheroids due to limited diffusion of nutrients and reduced clearance of metabolic waste [4]. This means that in a spheroid with a diameter of at least 500 µm, cells can be divided in three different regions (Figure 1):

- an external region formed by cells that are actively proliferating (*"Proliferation zone"*);

- an intermediate region formed by quiescent cells (*“Quiescent zone”*);
- an internal region formed by necrotic cells (*“Necrotic core”*).

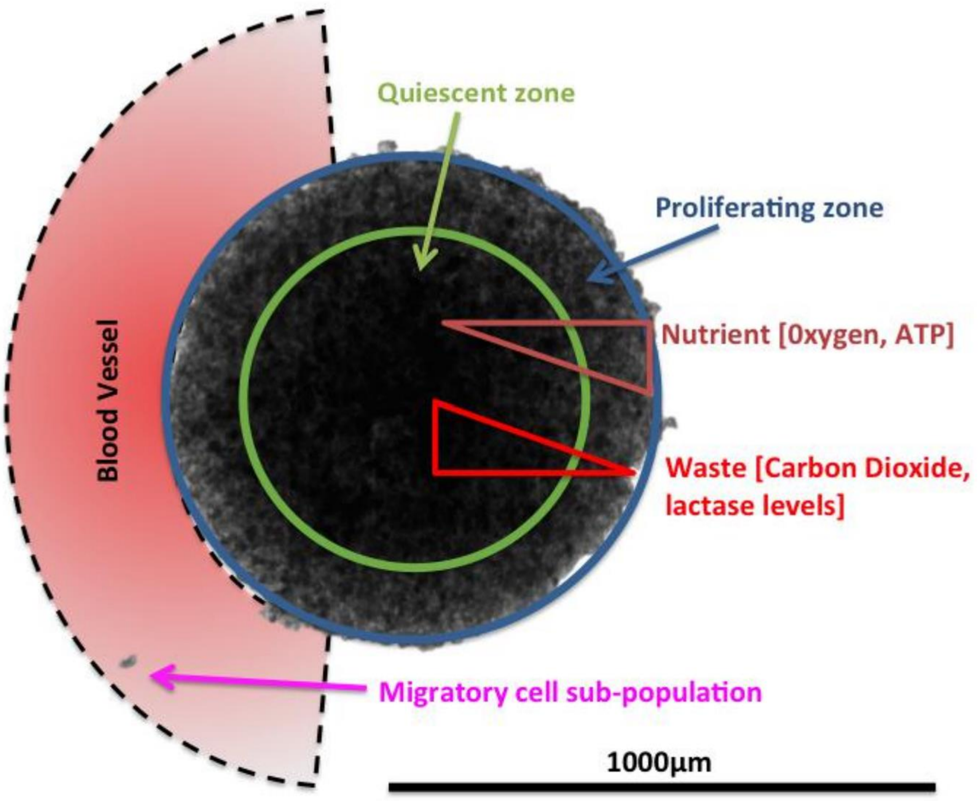


Figure 1: Internal structure of a spheroid, courtesy by [4].

There are several ways through which it is possible to generate spheroids, such as the hanging drop method, the agitation-based method, magnetic levitation method and spinner flask and gyratory rotation systems.

1.1 Hanging drop method

The hanging drop method is a widely used technique for generating spheroids. This method provides a simple and effective way to create spheroids without the need for specialized equipment or complex procedures.

The process begins by generating cell suspension drops of 10 µL. This cell suspensions are carefully placed on the inverted lid of a Petri dish, and, after the dish is filled with a small volume of sterile water or culture medium, the lid is turned upside down. surface tension of the cell suspension makes the droplet hang from the attached surface and gravity will drag the cells to the bottom of the droplet. As time progresses, the cells within the droplet start to aggregate and self-assemble. The hanging drop environment creates a unique microenvironment that promotes cell-cell interactions and the formation of spheroids. The absence of a solid surface prevents cell adhesion and encourages the cells to interact with each other instead. This interaction leads to the gradual formation of a compact, multicellular spheroid.

One of the key advantages of the hanging drop method is its simplicity. The setup requires minimal equipment and materials, making it accessible to researchers with limited resources. Additionally, the hanging drop method allows for the generation of a single spheroid per droplet, which is advantageous for studying individual spheroids or performing high-throughput experiments.

Furthermore, the hanging drop method offers control over spheroid size and cell composition. By adjusting the number of cells in the droplet, researchers can regulate the size of the resulting spheroid. Moreover, different cell types or combinations can be used in separate droplets, enabling the creation of heterogeneous spheroids that mimic specific tissue structures or disease models.

The hanging drop method also provides a dynamic environment for spheroid formation. The hanging droplet allows for the exchange of nutrients and waste products with the surrounding medium, which supports cell viability and function within the spheroid. The exposure to oxygen and nutrients in the culture medium promotes cellular metabolism and differentiation, closely resembling the physiological conditions found *in vivo*.

Despite its advantages, the hanging drop method does have limitations. The manual handling of droplets can be time-consuming and challenging, particularly when working with large numbers of spheroids. The technique is also not well suited for long-term culture or large-scale production of spheroids.

In conclusion, the hanging drop method is a straightforward and efficient approach for generating spheroids. It provides control over spheroid size and composition, offers a dynamic environment for cellular interactions, and requires minimal resources. While it may not be suitable for all applications, the hanging drop method remains a valuable tool in the fields of tissue engineering, drug discovery, and regenerative medicine, enabling researchers to study complex cellular interactions and model various diseases in a three-dimensional context [5].

1.2 Agitation-based methods

These methods involve culturing cells in a suspension that is subjected to continuous agitation, promoting cell-cell interactions and the formation of compact spheroids.

There are several types of agitation-based methods, including the use of orbital shakers, bioreactors, gyratory rotation, and spinner flasks. These systems provide a controlled and reproducible environment for spheroid formation. Cells are suspended in a culture medium within the agitation apparatus, and the constant motion prevents the cells from adhering to the surface of the vessel.

Orbital shakers are one of the simplest forms of agitation-based methods. The culture flasks or plates are placed on a shaker platform that moves in a circular motion. The shaking motion ensures an even distribution of cells and nutrients, facilitating the aggregation of cells into spheroids. Orbital shakers are commonly used in laboratories due to their ease of use and cost-effectiveness.

Bioreactors offer a more advanced approach to agitation-based methods. These systems provide precise control over parameters such as agitation speed, oxygen supply, pH, and temperature. Bioreactors often feature impellers or paddles that create a turbulent flow within the culture medium, promoting cell aggregation and spheroid formation. The enhanced control and scalability of bioreactors make them suitable for large-scale production of spheroids.

Spinner flask culture has been the most extensively used technique to culture large quantities of spheroids. Cells are cultivated as monolayers at first until they reach the exponential growth phase. They are then removed and placed in the spinner flask, where they develop in liquid media as a suspension culture. A cylindrical vessel is equipped with a central impeller that rotates at a constant speed, inducing a circular flow within the culture medium. The rotation of the impeller generates shear forces that prevent cell attachment to the flask's surface, allowing cells to aggregate and form spheroids. Spinner flasks are particularly useful when culturing cells that are sensitive to shear stress.

The gyratory rotation system is another technology that operates on the same principles. Erlenmeyer flasks are filled with cells that have previously grown in a monolayer. Cell-seeded culture vessels are placed on a gyratory shaker that tilts and rotates in a controlled manner. Gyratory motion generates a three-dimensional environment in which cells are subjected to dynamic forces such as gravitational and centrifugal forces. These forces encourage cell aggregation and compaction, resulting in the creation of spheroids. Spheroids can form in a matter of hours.

Agitation-based methods offer several advantages for spheroid generation. They procedures provide dynamic settings that closely resemble the *in vivo* conditions required for spheroid formation. These technologies allow for the investigation of cellular behaviour in three dimensions, increasing the relevance and physiological value of the created spheroids. Both approaches enable the inclusion of multiple cell types, resulting in heterotypic spheroids that closely mirror the cellular variety present in real tissues. This skill is essential for investigating complicated cellular relationships and tissue-specific functions. Researchers can reproduce the microenvironment and cellular interplay observed in physiological tissues by merging different cell types within a rotating or gyrating environment. The continuous motion prevents cell adhesion, enabling the formation of spheroids with high cell viability. The controlled agitation also promotes nutrient and oxygen distribution, supporting cell growth and metabolism within the spheroid. Additionally, these methods allow for scalability, making them suitable for generating large quantities of spheroids for various applications.

However, there are certain limitations associated with agitation-based methods. The mechanical forces generated during agitation can induce shear stress, which may affect cellular behaviour and functionality. Certain cell types may be more sensitive to shear stress and may require optimization of agitation parameters. Spheroids produced using spinner flask/gyratory rotation systems and liquid overlay techniques, however, are heterogeneous in size and shape. Because homogenous spheroids are preferred for reproducibility and dependability in high-throughput tests, they must be manually sorted [6].

In conclusion, agitation-based methods are widely used techniques for generating spheroids. They provide a controlled environment that promotes cell aggregation and spheroid formation. These methods offer advantages such as high cell viability, scalability, and uniform nutrient distribution. While there are limitations to consider, agitation-based methods remain valuable tools in tissue engineering, drug discovery, and regenerative medicine research, enabling the study of cellular behavior in a three-dimensional context that closely resembles native tissue environments.

To standardize the analysis of chemotherapy drugs, it is fundamental to extract several bio-morphological features from the spheroids to analyse the changes induced by the treatment. between them, the calculation of the volume of the spheroids in particular is quite significant for obtaining an accurate evaluation of the efficiency of the treatment being evaluated [7].

1.3 Magnetic levitation method

The magnetic levitation technique is a novel and promising method for creating spheroids. This method employs magnetic nanoparticles that are combined with cells and exposed to a regulated magnetic field, causing cell aggregation and the development of spheroids. Magnetic levitation has several distinct advantages over conventional spheroid formation processes, including precise control over spheroid size and composition and reproducibility.

Basically, the hydrogel is dispersed over cells and the mixture is incubated. Then a washing step removes the non-interacting hydrogel fragments. Fractions of phage, gold and MIO nanoparticles enter cells or remain membrane bound. Then a magnetic field is applied that causes the cells to rise to the air-medium interface and finally, after 12 hours of levitation, the characteristic multicellular structure forms [8].

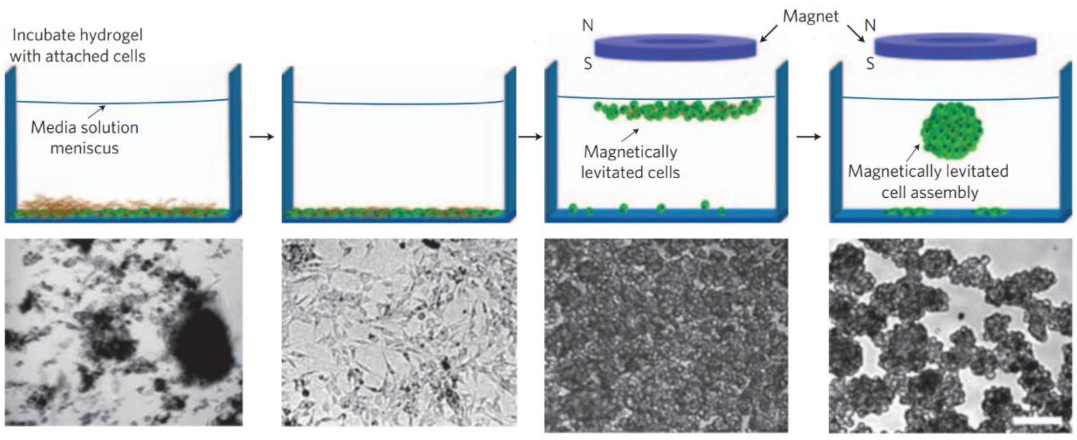

Incubate hydrogel with attached cells
Media solution meniscus
N
S
Magnet
N
S
Magnetically levitated cells
Magnetically levitated cell assembly

Figure 2: 3D spheroid generation with magnetic-based levitation method. The first row shows the general strategy, while the second shows the corresponding microscopy image at each stage. Courtesy by [8].

The capacity to accurately control the size of the resulting spheroids is a fundamental advantage of magnetic levitation. Researchers can modify the magnetic forces acting on nanoparticles and so regulate cell aggregation and compaction by varying the strength and duration of the applied magnetic field. This ability to adjust spheroid size is critical for replicating distinct tissue topologies and designing spheroids to satisfy the needs of various applications, such as drug screening and tissue engineering.

Another key advantage of magnetic levitation is its reproducibility. The magnetic field exerts a consistent and predictable force on the magnetic nanoparticles, resulting in consistent and dependable spheroid formation across several experiments. This reproducibility enables researchers to compare and confirm data from various investigations, increasing the trustworthiness and credibility of the created spheroids as model systems. Furthermore, the magnetic levitation technology allows for the production of complicated spheroids. Magnetic nanoparticles can be combined with different cell types to create heterotypic spheroids that closely match the cellular variety present in native tissues. This ability is invaluable for investigating complicated cellular interactions, such as those that occur in organs or during disease development. Additionally, by incorporating patient-derived cells, the magnetic levitation method can generate personalized spheroids that offer a more accurate representation of an individual's disease or response to therapy.

1.4 Scaffold-Based systems

Scaffold-based systems have emerged as a key component in tissue engineering approaches, providing a structural framework that supports cell adhesion, proliferation, and organization. These systems have revolutionized the field by enabling the generation of tissue-like constructs that mimic the native extracellular matrix (ECM) and recapitulate the complex microenvironment required for tissue development and regeneration. They use synthetic or naturally derived biomaterials that simulates the

ECM in attempt to redirect the growth of the cells in a specific spatial configuration and maintaining their function.

In order to be able to do so, the material should:

- have surface promoting attachment, proliferation and differentiation;
- be highly porous, to allow cell growth and turnover of waste and nutrient materials;
- be biodegradable at a rate that can be controlled and that matches the cell growth;
- provide all the chemical and mechanical inputs that the *in vivo* tissue normally receive;

They have been used for the long-term generation of spheroids of many cell types. In particular, this kind of generation method provides the right conditions for simulating the mechanical stress that the cells would undergo in in vivo tissues, in order to produce a more natural phenotype.

The choice of the scaffold material and its compositions are fundamental. They are in fact known to affect the behaviour and morphology of the generated spheroids. Developing a scaffold material that can simulate the ECM is difficult due to the complexity of its bio-mechanical characteristics. Over the years, scaffold-based systems have witnessed remarkable advancements, primarily driven by advancements in biomaterials and fabrication techniques. The development of biocompatible materials, including hydrogels, synthetic polymers, and decellularized ECMs, has expanded the repertoire of scaffolds available for tissue engineering. These materials offer tunable properties such as mechanical strength, degradation kinetics, and bioactivity, allowing researchers to tailor the scaffold's characteristics to specific tissue types and applications. Additionally, advances in scaffold fabrication techniques, such as 3D printing and electrospinning, have facilitated precise control over scaffold architecture, pore size, and distribution, enabling the creation of complex tissue structures with high fidelity.

One of the most common ways to produce spheroids through a scaffold-based system is to co-culture two or more types of cells on the same scaffold system. This can produce highly sophisticated models that are able to better mimic in vivo conditions, including matrix remodelling under mechanical stress.

In conclusion, scaffold-based systems have emerged as a critical component in tissue engineering, offering a versatile platform for the generation of complex tissue constructs. The advancements in biomaterials and fabrication techniques have expanded the possibilities for scaffold design and customization. These systems have found applications in organoid and spheroid generation, tissue regeneration, and neural tissue engineering. Future directions focus on the development of bioactive and smart scaffolds, as well as the integration of scaffold-based systems with other cutting-edge technologies [6].

2 Radiomics: insight from medical images through quantitative analysis

Radiomics is a rapidly evolving field that aims to extract quantitative features from medical images and analyse them to provide valuable information for diagnosis, prognosis, and treatment response assessment. It involves the application of advanced image processing techniques, machine learning, and statistical modelling to extract and analyse a large number of imaging features from medical images.

Radiomics is based on the principle that medical images contain a wealth of information beyond what is visible to the naked eye. These images capture detailed spatial and textural patterns that can be quantified and analysed to reveal insights about underlying tissue characteristics, tumor heterogeneity, and disease progression. Radiomics leverages pixel- or voxel-level intensity values, as well as spatial relationships and texture patterns, to derive meaningful and reproducible features.

The workflow of radiomics analysis typically involves several essential steps. First, medical images are acquired using standardized protocols to ensure consistency and quality. Next, preprocessing techniques are employed to address potential noise, artifacts, and variations in image acquisition. Image segmentation techniques then delineate regions of interest (ROIs), such as tumors or organs, establishing the target for subsequent analysis. It is within these ROIs that radiomic features are extracted, encompassing a wide range of characteristics. These features may include intensity-based measurements, such as mean or standard deviation, shape descriptors, texture features that capture patterns within the image, and spatial relationships between neighbouring pixels or voxels.

However, the extraction of a large number of features can introduce challenges related to high dimensionality and potential overfitting. To address this, feature selection methods are applied to identify the most relevant and informative features for subsequent analysis. Once the features are selected, they serve as the basis for developing predictive models or classifiers. Machine learning algorithms or statistical models are then trained using these features, enabling the prediction of clinical outcomes, classification of tissue types, or assessment of treatment response.

Radiomics holds tremendous promise across various medical domains. It can be used to capture tissue and lesion qualities like form and heterogeneity, as well as changes in those properties over time, such as during treatment or surveillance. In oncology, it has shown significant potential for tumor characterization, treatment response prediction, and patient prognosis.

Genomic studies have shown that tumor heterogeneity is a predictive factor for survival and a barrier to cancer control. According to research, radiomic characteristics are closely associated with cellular heterogeneity indices. While biopsies typically capture heterogeneity within a limited section of a tumor and at a single anatomic site, radiomics captures heterogeneity across the total tumor volume. Unsurprisingly, radiomic characteristics are also linked to tumor aggressiveness. By capturing the heterogeneity within tumors, radiomics offers insights into their genetic and phenotypic diversity, aiding in treatment planning and personalization. Furthermore, radiomics has found applications in neuroimaging, cardiovascular imaging, lung disease assessment, and musculoskeletal disorders, among others. Radiomic features have also been related to genomic, transcriptomic, or proteomic properties and have been proposed to predict clinical endpoints such as survival and treatment response. Although individual radiomic features may correlate with genomic data or clinical outcomes, radiomics' impact is increased when the wealth of information it provides—typically hundreds of features, a fraction of which will contribute to a disease-specific radiomic signature—is processed using machine learning techniques. Second, radiomic data are mineable, meaning that in sufficiently large datasets, they may be used to discover previously unknown markers and patterns of disease evolution, progression, and treatment response. This so-called population-imaging approach may either use unstructured data from different modalities (*e.g.*, PET, CT, and MRI) acquired for a specific but possibly unrelated diagnostic purpose in broadly defined groups or may use—as in the German National MRI Cohort Study—a single imaging test in a large cohort for a multicentric longitudinal observational study. Such radiomic data can be combined with clinical, laboratory, histologic, genomic, or other data using unsupervised machine learning.

Despite its tremendous potential, radiomics is not without challenges. Standardization and reproducibility across different imaging platforms and institutions remain critical concerns. Variations in imaging protocols, image quality, and segmentation techniques can introduce inconsistencies in feature extraction and impact the reliability of radiomics models. Moreover, the clinical translation and validation of radiomics require extensive studies and validation cohorts to establish their true clinical utility and value.

There are five main groups of parameters researched in radiomics:

- **Histogram features**: Gray-level mean, calculated as the sum of all the grey level values of the pixels over the total number of pixels,

  $$GL_{mean} = \frac{\sum(GL\_i)}{N}$$

  Where GL_i is the grey level of each pixel and N is the total number of pixels. Maximum grey level value, minimum grey level value, variance, that can be calculated as:

  $$V = \frac{\sum GL_i - GL_{mean}}{N - 1}$$

  and percentiles are the most basic statistical descriptors based on the global gray-level histogram. These features are referred to as first-order features since they are based on single-pixel or single-voxel analysis. Skewness and kurtosis are more sophisticated features that describe the shape of a data distribution's intensity distribution. The skewness reflects asymmetry of the data distribution curve to the left (negative skew, below the mean) or right (positive skew, above the mean), and it can be calculated as:

  $$\text{Skewness} = \frac{\sum(GL_i - GL_{mean})^3/\text{N}}{V^3}$$

The skewness from a normal distribution is zero. On the other hand, kurtosis reflects the tailedness of a data distribution relative to a gaussian distribution due to outliers.

$$\text{Kurtosis} = \frac{\sum(GL_i - GL_{mean})^4/\mathrm{N}}{V^4}$$

Other characteristics include histogram entropy and homogeneity.

- **Texture features:** The examination of the absolute gradient, which indicates the degree of abruptness of gray-level intensity fluctuation across a picture, is one of the simplest approaches to radiomic texture description. The gradient is at its greatest when two neighboring pixels are one white and one black, however the absolute gradient is 0 when both pixels are black or white. The gradient's direction is unimportant; the only crucial parameter is its intensity. Mean, variance, skewness, and kurtosis, like histogram features, can be calculated as matrices. There are numerous matrices, also known as second-order gray-level histograms, that can be used to compute entropy, inhomogeneity, the spatial distribution of consecutive pixels, and many other things.
- **Model-based features:** Model-based analyses seek to characterize objects or forms by interpreting spatial gray-level information. The ROI is fitted with a parameterized model of texture production, and the predicted parameters are employed as radiomic features. An example of a model-based method is the autoregressive model, which is based on the assumption that a pixel's gray level is a weighted sum of the gray levels of four nearby pixels: the pixel to its left, top left, top, and top right; and sigma, which contains information on the variance of the minimum prediction error. Fractal analysis also produces properties useful in radiomics, specifically fractal dimension, which indicates the rate of addition of structural detail with

increasing magnification, size, or resolution and so serves as a measure of complexity. Inhomogeneity is reflected by lacunarity, a trait that measures the lack of rotational or translational invariance.

- **Transform-based features:** Gray-level patterns are analyzed in a separate space using transform-based approaches such as Fourier, Gabor, and Haar wavelet transforms. For example, the discrete Haar wavelet transform examines the frequency content of an image at various scales. A pair of quadrature mirror filters, a high-pass and a low-pass filter, can be used to decompose a picture into wavelets. While the high-pass filter amplifies fluctuations in gray level and thus emphasizes image details, the low-pass filter smooths the image in terms of gray level and thus removes image features. Following signal decomposition, a collection of spatially oriented frequency channels is available for describing local visual fluctuation. The energies contained inside the frequency channels are then utilized to create features. Both directions of high-pass filtering capture diagonal details; high-pass filtering followed by low-pass filtering captures vertical edges; low-pass filtering followed by high-pass filtering captures horizontal edges; and low-pass filtering in both directions captures the lowest frequencies at different scales. Notably, wavelet transformation can be utilized for more than only radiomic feature generation; it can also be employed for picture segmentation or as a pretreatment step for texture analysis.
- **Shape-based features:** The geometric qualities of ROIs are described by shape-based attributes. Many shape-based features, such as 2D and 3D diameters, axes, and their ratios, are theoretically simpler than other radiomic features. Surface- and volume-based techniques based on meshes (small polygons like triangles and tetrahedrons) are more sophisticated. Compactness and sphericity, which define how

a ROI differs from a circle (for 2D analyses) or a sphere (for 3D analyses), and density, which rely on the building of a minimum-oriented bounding box (or rectangle for 2D analyses) encompassing the ROI, are two examples of this kind of features.

Summarizing, radiomics is an advanced image analysis tool with the potential to revolutionize precision medicine. Radiomic characteristics not only correlate with genomic data, but they may also provide complementary information regarding tumor heterogeneity across the entire tumor volume, potentially improving survival prediction and proving valuable for patient classification. With a lengthy history of delivering quantitative biologic data, radiomics could be the next logical step in nuclear medicine's progression, not only as a clinical decision-making tool but also as a research tool to find novel molecular disease pathways. The creation and rigorous adherence to standardized picture acquisition and reconstruction techniques, on the other hand, are critical [9].

3 Deep learning with CNNs

Machine learning and deep learning are two areas of artificial intelligence that gained an increasing amount of attention and have influenced several industries. Machine learning is a branch of artificial intelligence that focuses on creating algorithms and models that allow computers to learn from data and make predictions or judgments without being explicitly programmed. Deep learning, on the other hand, is a subset of machine learning that entails training multiple-layer artificial neural networks to extract high-level features from raw data [10].

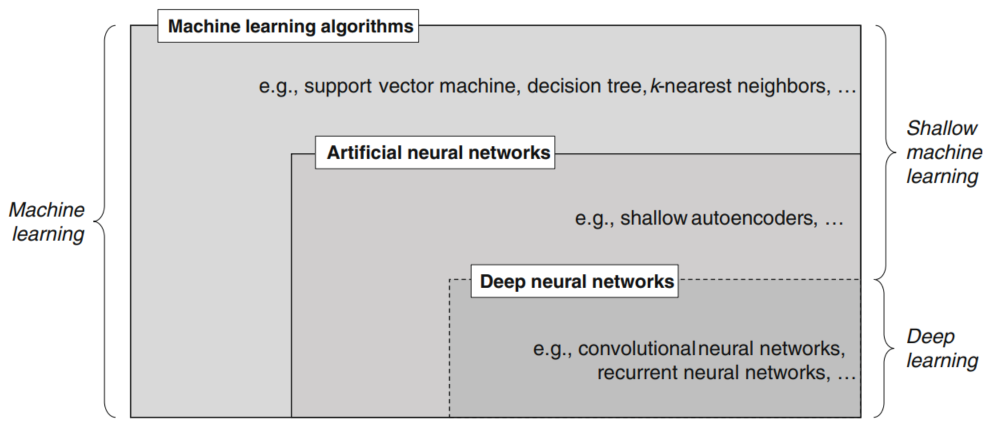

Machine learning algorithms
e.g., support vector machine, decision tree, k-nearest neighbors, ...
Artificial neural networks
e.g., shallow autoencoders, ...
Deep neural networks
e.g., convolutional neural networks, recurrent neural networks, ...
Machine learning
Shallow machine learning
Deep learning

Figure 3:Venn diagram of machine learning classes. Courtesy by [10].

Machine learning describes those situations in which a computer program's performance can improve with time and experience regarding some class of tasks and features measures. Its aim is to automate the task of analytical model building to perform tasks such as object detection or language translation. This is achieved by applying algorithms iteratively that learn from a specific training dataset, allowing the code to find complex patterns without being directly programmed to do so. This technique is applied primarily to high dimensional tasks such as classification, clustering, etc. Based on the needs of the particular application the algorithm is engineered to do, it is possible to distinguish between three types of machine learning:

- **Supervised learning:** In supervised learning, a dataset is provided consisting of input features and corresponding target outputs, or labels. The goal is to train a model that can accurately map the input features to the correct target outputs. The process of supervised learning involves two main steps: training and inference. During the training phase, the model learns the underlying patterns and relationships in the data by optimizing its parameters, or weights. This optimization is achieved by minimizing a loss or cost function, which quantifies the discrepancy between the model's predictions and the actual labels in the training data. Once the model is trained, it can be used for inference on new, unseen data. Given the input features of a new instance, the model applies the learned mapping to make predictions or classifications. The performance of the supervised learning model is evaluated using evaluation metrics such as accuracy, precision, recall, F1 score, or area under the curve (AUC), depending on the specific task and domain.
- **Unsupervised learning:** Unsupervised learning is a branch of machine learning where models are trained to discover patterns and structures in unlabelled data. Unlike supervised learning, unsupervised learning algorithms do not rely on labelled examples or target outputs. Instead, they aim to extract meaningful information and representations directly from the input data. The main objective of unsupervised learning is to find hidden patterns, group similar data points, or

reduce the dimensionality of the data without any explicit guidance or labels. Common techniques used in unsupervised learning include clustering, dimensionality reduction, and generative modelling.

- **Reinforcement learning:** In reinforcement learning, an agent interacts with an environment, observes its current state, takes actions, and receives feedback in the form of rewards or punishments. Unlike supervised and unsupervised learning, reinforcement learning does not rely on labeled or unlabeled data. The objective of the agent is to learn a policy, a mapping from states to actions, that maximizes its cumulative rewards over time. The agent learns through a process of exploration and exploitation, trying different actions to discover the optimal strategy. Reinforcement learning algorithms use various techniques to balance exploration and exploitation. One popular method is Q-learning, which uses a value function called the Q-function to estimate the expected cumulative reward for taking a specific action in a given state. By updating the Q-values based on the rewards received and future expected rewards, the agent gradually improves its policy.

Depending on the learning task, the field offers various classes of ML algorithms, each of which comes in multiple specifications and variants, including regression models, instance-based algorithms, decision trees, Bayesian methods, and ANNs.

An Artificial Neural Network (ANN) is defined as a computational processing system that is significantly inspired by how biological nervous systems (such as the human brain) works. ANNs are primarily composed of a large number of interconnected computational nodes (referred to as neurons), which collaborate in a distributed method to collectively learn from input and optimize their ultimate output. The basic structure of a ANN can be modelled as shown in Figure 4.

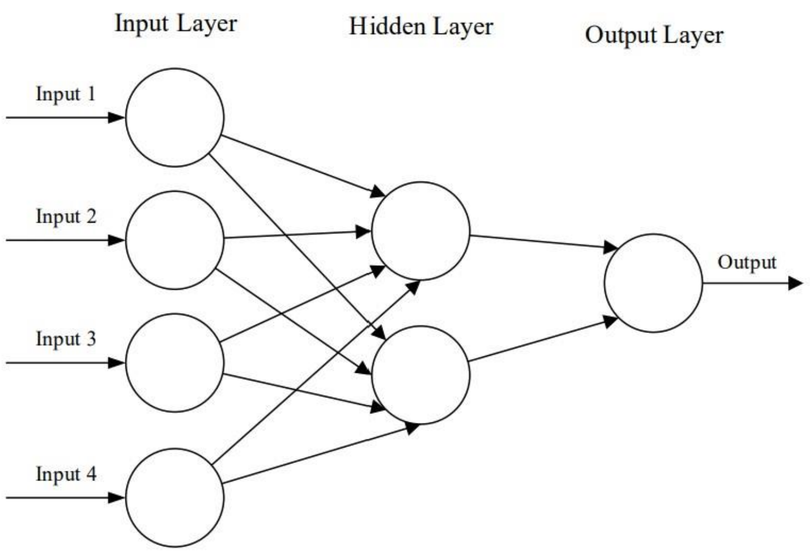


Figure 4: Scheme of an artificial neural network. Image courtesy by [11].

Each connection between neurons, like synapses in the brain, transmits signals whose strength can be amplified or decreased by a weight that is continuously modified during the learning process. Signals are only processed by following neurons if a particular threshold, specified by an activation function, is exceeded. Neurons are typically organized into networks with multiple layers. Typically, an input layer accepts data input, and an output layer provides the final result. There are zero or more hidden layers in between that oversee learning a non-linear mapping between input and output. The learning algorithm cannot learn the number of layers and neurons, as well as other property options such as learning rate or activation function. They are the hyperparameters of a model and must be set manually or calculated by an optimization method. Deep neural networks are often made up of multiple hidden layers that are arranged in highly nested network designs. Furthermore, unlike ordinary ANNs, they typically incorporate advanced neurons. That means that, rather than a basic activation function, they may employ complex operations (*e.g.*, convolutions) or numerous activations in a single neuron. Deep neural networks can be fed raw input data and automatically identify the representation required for the appropriate learning task because of these properties. This is the network's primary capability, often known as deep learning.

Machine learning algorithms and very simple ANNs can be gathered under the term of shallow machine learning since they cannot reach the capabilities of deep learning neural networks, even if there is not a clear differentiation between this two concepts.

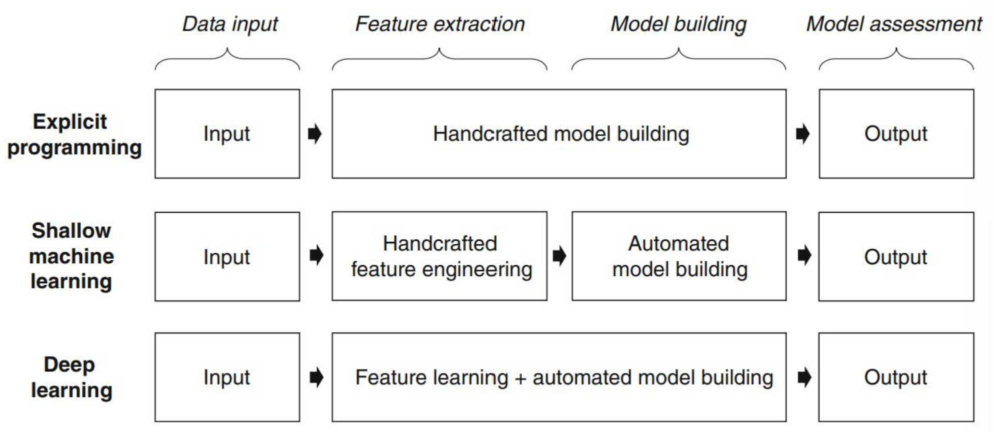


Figure 5: Process of analytical building of different models. Courtesy by [11].

The deep learning neural networks can be categorized in recursive neural networks (RvNNs), recurrent neural networks (RNNs) and convolutional neural networks (CNNs).

The most famous and commonly employed ones are the Convolutional Neural Networks (CNNs) [11]. CNNs are similar to regular ANNs in that they are made up of ’neurons’ that optimize themselves through learning. Each neuron will continue to accept input and conduct an operation (such as a scalar product followed by a non-linear function), which is the same foundation of many ANNs. The entire network will still express a single perceptual scoring function from the input raw picture vectors to the final output of the class score (the weight). The final layer will include loss functions related to the classes, as well as all the standard tips and tricks created for classic ANNs that still apply. The main improvement of these types of networks compared to the others is that they can automatically identify the relevant features without any human supervision. They have been used in various field such as computer vision, speech processing, face recognition, and pattern recognition within images. This means that it is possible to encode image-specific features into the architecture, making the network more suited for image-focused tasks - whilst further reducing the parameters required to set up the model. They can handle jobs involving datasets with spatial relationships in which the columns and rows are not interchangeable (for example, image data). Their network design is made up of a succession of steps that allow for hierarchical feature learning based on the modelling job. When it comes to object recognition in photos, for example, the initial few layers of the network are in charge of extracting basic properties such as edges and corners. In the final layers, these are increasingly aggregated into more complex characteristics that resemble the actual things of interest, such as animals, buildings, or cars. Following that, the auto-generated features are utilized to forecast objects of interest in new photos. An example of CNN architecture for image classification is illustrated in Figure 6.

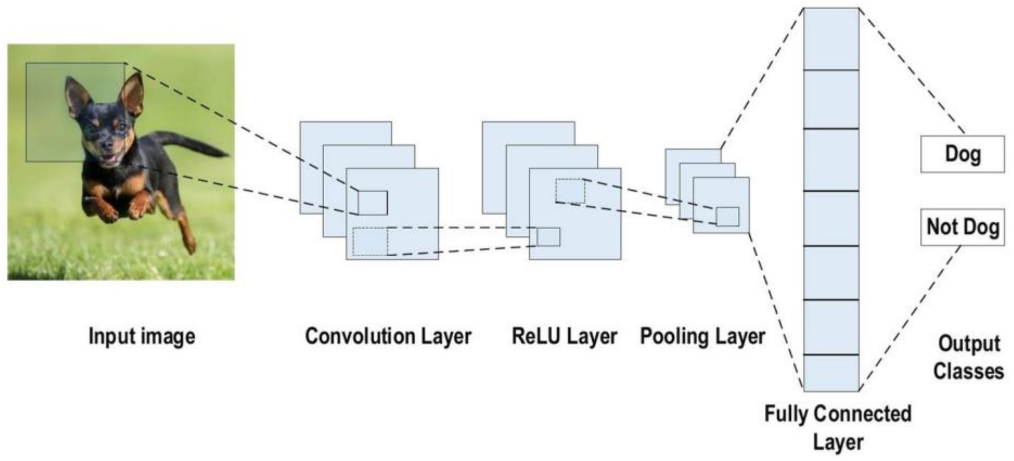


Figure 6: Example f CNN architecture for image classification. Courtesy by [12].

Every layer of the CNN model's input is structured in three dimensions: height, width, and depth. The depth is also known as the channel number. In an RGB image, for example, the depth is three because there are three colour channels. Each convolution layer contains numerous kernels that must have three dimensions, similar to the input picture; the convolution layer computes a dot product between its input and the weights. The following step is to down sample each feature map in the sub-sampling layers. This reduces the network parameters, which speeds up the training process and allows for the treatment of the overfitting issue. Finally, like in a conventional neural network, the fully connected layers receive the mid- and low-level characteristics and generate the high-level abstraction, which represents the final-stage layers. The classification scores are calculated based on the ending. Every score for a given instance represents the probability of a specific class.

The main benefit of using CNNs over other traditional models of neural networks are the weight sharing feature, that reduces the number of trainable parameters helping to enhance generalization and avoid overfitting; The ability to learn at the same time the feature extraction layers and the classification layer causes the output trained model to be highly organized and reliant on the extracted features; Large-scale implementation of networks is easier with CNNs than with the other neural networks. All of these factors contributed to the decision to use CNNs in this project.

The various CNNs architectures consist of different layers, also known as multi-building blocks. The more common ones are the convolutional layers, pooling layers, reLu layers and fully connected layers [12].

- **Convolutional Layers:** they are the most significant component in a CNN architecture. They consist in a series of convolutional filters, also known as kernels, in which the input image, used as an N-dimensional matrix, is convolved to generate the output feature map. The kernel is defined as a grid of discrete numbers called kernel weight. There are several ways in which it is possible to initialize these weights. They are then adjusted every training epoch in such way that the kernel learns to extract the significant feature inside the images. The kernel, defined as a matrix with lower dimensions than the input, slides over the whole image and the dot product between the input and the kernel is defined. The corresponding values are multiplied and then summed to create a single scalar value. The process is repeated until no more sliding of the kernel is possible.

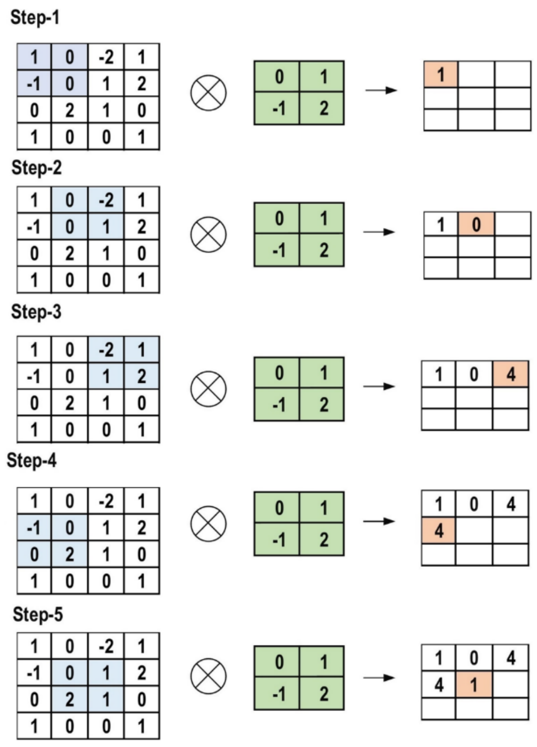


Figure 7: example of calculation performed at every step of convolutional layer. Courtesy by [12].

The general mathematical formula describing the convolution operation is:

$$[M] * [K] = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} m_{(m-1)(n-1)} k_{(1+i)(1+j)}$$

Where M is the portion of the image on which the convolution is performed, represented in blue in Figure 7, K is the kernel and m, and n are the dimensions of the kernel.

- **Pooling Layers:** these layers are responsible of the sub-sampling of the features maps, that are generated as a result of the convolutional operations. They shrink large feature maps into smaller ones. At the same time, it maintains the same level of information in every step of the pooling process. Similarly to the convolutional layers, also here there is an initially assigned kernel that is then updated every cycle. The most common one is the max pooling layer, which chooses the higher value present in the portion of the matrix that is being analysed by the kernel.

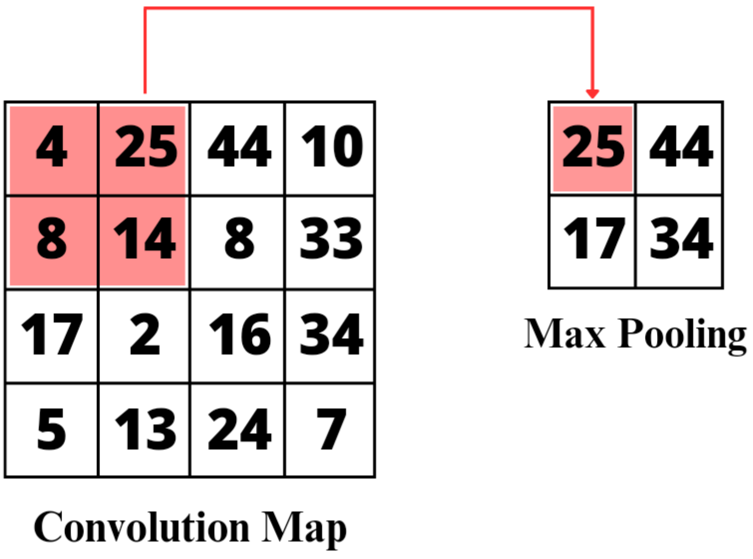


Figure 8: Max pooling layer representation.

- **ReLU layers:** A ReLU (Rectified Linear Unit) layer is a common activation function used in neural networks. For a given input, the output of the ReLU is the maximum between 0 and the input. This means that only positive inputs can pass this layer. The ReLU activation function introduces non-linearity into the neural network, enabling it to learn complex patterns and make nonlinear transformations. One of the key advantages of the ReLU function is that it helps mitigate the vanishing gradient problem that can occur during backpropagation, a process used to train neural networks. The ReLU function allows gradients to flow freely for positive values, avoiding the saturation of gradients that can hinder the training process. The ReLU layer is typically applied after a linear transformation, such as a fully connected layer or a convolutional layer, in a neural network. By introducing sparsity in the network, ReLU layers help create sparse representations, which can improve the efficiency of computation and reduce overfitting.

- **Fully connected layers:** This layer is located at the end of each CNN architecture. Inside this layer, each neuron is connected to all neurons of the previous layer, the so-called Fully Connected (FC) approach. It is utilized as the CNN classifier. The input for the FC layer comes from the last pooling or convolutional layer. The input is in the form of a vector, which is created from the feature maps after fattening. The output of the FC layer represents the final CNN output.

Over the last 10 years, several CNN architectures have been presented. Model architecture is a critical factor in improving the performance of different applications. Various modifications have been achieved in CNN architecture from 1989 until today. Such modifications include structural reformulation, regularization, parameter optimizations, etc. For this particular projects, four different CNNs were implemented using the deep learning tool created by MATLAB: VGG16, VGG19, ResNet18, and ResNet50.

3.1 VGG networks

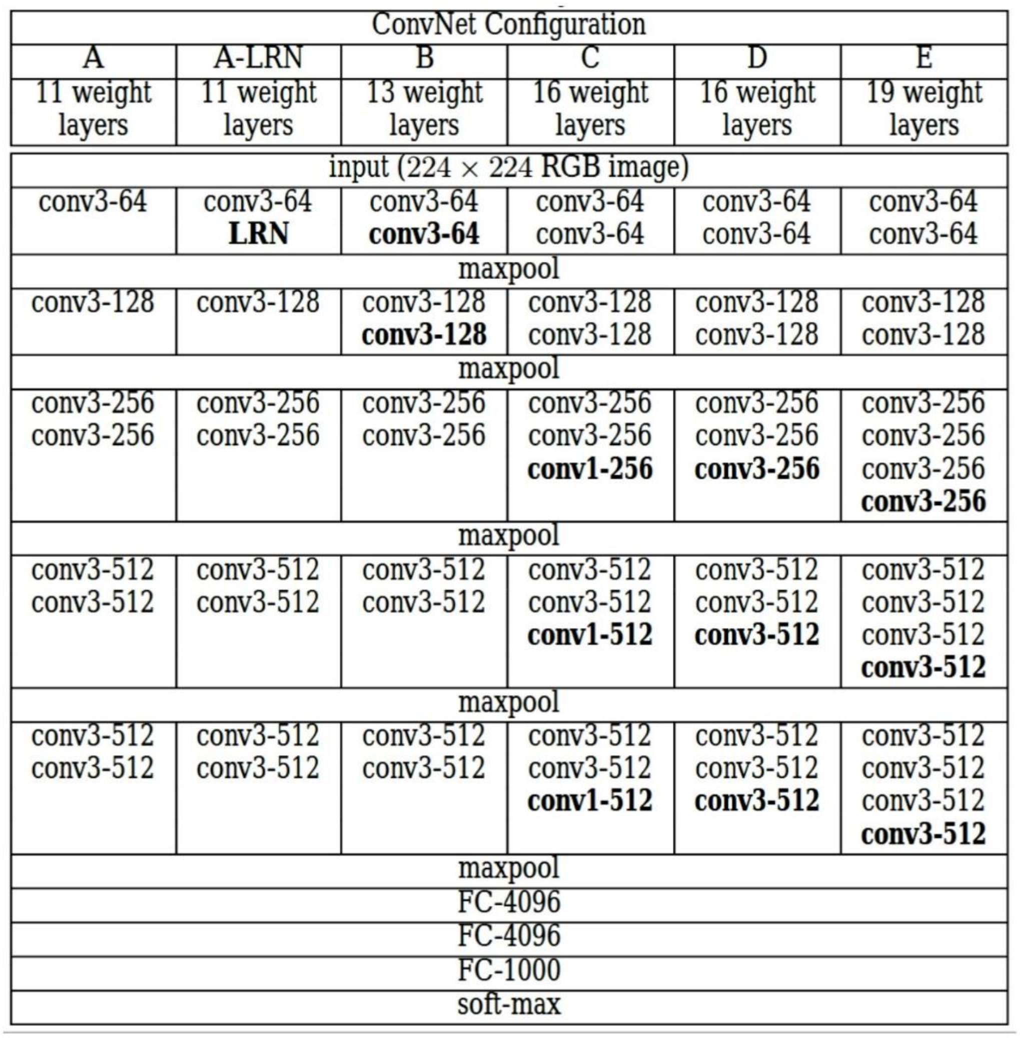
VGG stands for Visual Geometry Group; it is a standard deep CNN architecture with multiple layers [11]. The "deep" refers to the number of layers with VGG16 or VGG19 consisting of 16 and 19 convolutional layers respectively. The VGG architecture is the basis of ground-breaking object recognition models. Developed as a deep neural network, the VGG Net also surpasses baselines on many tasks and datasets beyond ImageNet. Moreover, it is now still one of the most popular image recognition architectures. It was based on an analysis of how to increase the depth of such networks. The network utilises small 3 x 3 filters. Otherwise, the network is characterized by its simplicity: the only other components being pooling layers, used to reduce the dimension of the feature map, and a fully connected layer, that are the last layers of the network, creating the final classification map. The VGG model, or VGGNet, can classify images into 1000 object categories, including keyboard, animals, pencil, mouse, etc. The model has an image input size of 224-by-224. The concept of the VGG16 and the VGG19 model is the same except that it supports 16 or 19 layers respectively. The "16" and "19" stand for the number of weight layers in the model (convolutional layers). This means that VGG19 has three more convolutional layers than VGG16.

The VGG16 consists of 13 convolutional layers and three fully connected layers.

- **Input:** The VGG Net takes in an image input size of 224×224. For the ImageNet competition, the creators of the model cropped out the centre 224×224 patch in each image to keep the input size of the image consistent.

- **Convolutional Layers:** VGG's convolutional layers leverage a minimal receptive field, *i.e.*, 3×3, the smallest possible size that still captures up/down and left/right. Moreover, there are also 1×1 convolution filters acting as a linear transformation of the input. This is followed by a ReLU unit, which is a huge innovation from *AlexNet* that reduces training time. ReLU stands for rectified linear unit activation function; it is a piecewise linear function that will output the input if positive; otherwise, the output is zero. The convolution stride is fixed at 1 pixel to keep the spatial resolution preserved after convolution (stride is the number of pixels shifts over the input matrix).
- **Hidden Layers:** All the hidden layers in the VGG network use ReLU. VGG does not usually leverage Local Response Normalization (LRN) as it increases memory consumption and training time. Moreover, it makes no improvements to overall accuracy.
- **Fully Connected Layers:** The VGGNet has three fully connected layers. Out of the three layers, the first two have 4096 channels each, and the third has 1000 channels, 1 for each class.

The VGG19 network has basically the same architecture but with three more deep layers, as shown in Figure 9 in column D and E.



ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 9: Architecture of different networks. Row E and D represent VGG16 and VGG19 respectively. Courtesy by [11].

3.2 ResNet networks

Residual Network (ResNet) is one of the famous deep learning models that was introduced by Shaoqing Ren, Kaiming He, Jian Sun, and Xiangyu Zhang in their paper. The paper name is *"Deep Residual Learning for Image Recognition"* [13]. The ResNet model is one of the popular and most successful deep learning models so far. The main reason for which these types of CNN were developed was to solve the so called *"vanishing/exploding gradients"* problem, which hampers convergence from the beginning. As the number of layers increases in CNN, the ability of the model to fit more complex functions also increases. Hence, more layers promise better performance. Now the question is, why shouldn't you use a VGGNet with more layers, such as VGG20, or VGG50, or VGG100? This is where the problem arises. The weights of a neural network are updated through the backpropagation algorithm, which makes a minor change to each weight so that the loss of the model decreases. But how does it occur? It updates each weight so that it takes a step in the direction along which the loss decreases. This is nothing but the gradient of this weight which can be found using the chain rule. However, as the gradient keeps flowing backward to the initial layers, the value keeps increasing by each local gradient. This results in the gradient becoming smaller and smaller, thereby making changes to the initial layers very small. This, in turn, increases the training time significantly. The solution to this problem, after which the networks are named, is the introduction of the so called deep residual learning framework.

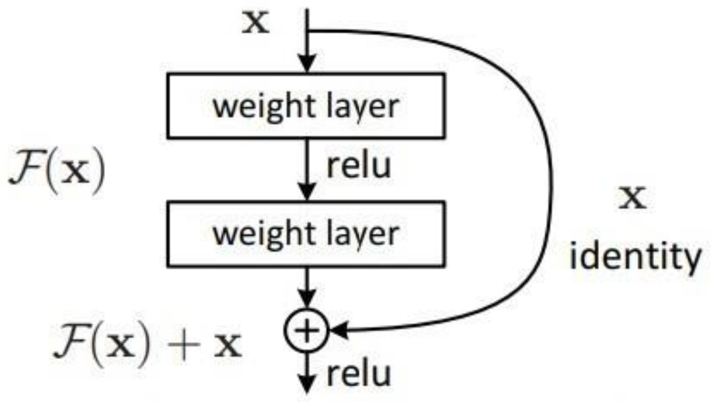
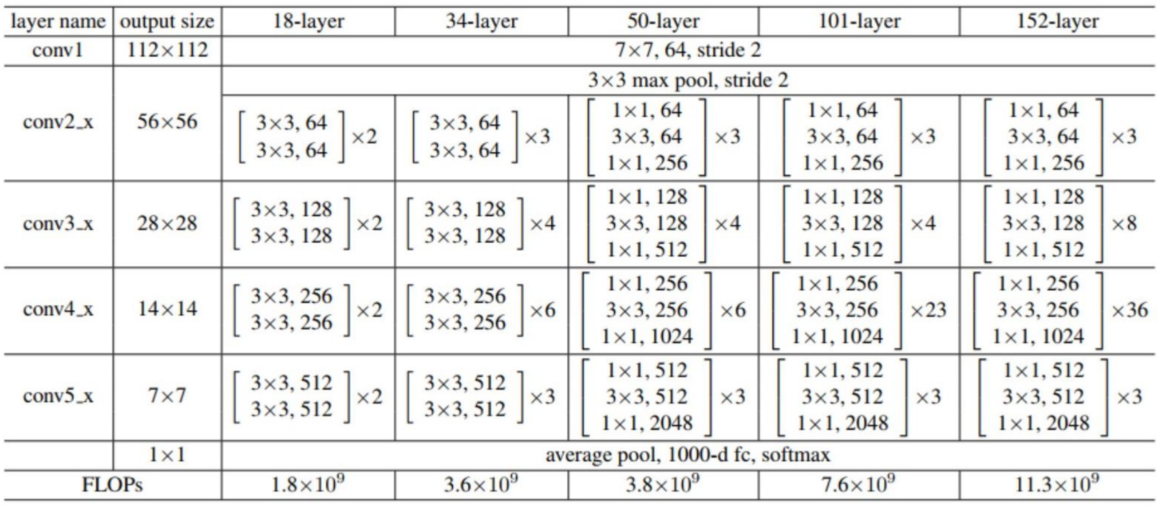

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 10: Residual block structure. Courtesy by [13].

Instead of hoping each few stacked layers directly fit a desired underlying mapping, these layers are set to explicitly fit a residual mapping. Formally, denoting the desired underlying mapping as H(x), the stacked nonlinear layers fit another mapping of F(x)≔H(x)x. The original mapping is recast into F(x)+x. It is hypothesized that it is easier to optimize the residual mapping than to optimize the original, unreferenced mapping. The formulation of F(x) +x can be realized by feed forward neural networks with *"shortcut connections"* as seen in Figure 10. Shortcut connections are those skipping one or more layers. In our case, the shortcut connections simply perform identity mapping, and their outputs are added to the outputs of the stacked layers. Identity shortcut connections add neither extra parameter nor computational complexity [14]. Compared to VGGNets, ResNets are less complex since they have fewer filters. ResNet does not allow the vanishing gradient problem to occur. The skip connections act as gradient superhighways, which allow the gradient to flow undisturbed. This is also one of the most important reasons why ResNet comes in versions like ResNet50, ResNet101, and ResNet152. The differences between the ResNet18 architecture, the ResNet50, and the ResNet101 one is only in the number of deep layer used to build the residual network, but the basic concept behind them remains the same as shown in Figure 11. Generally speaking, it is considered valid the concept that networks with a higher number of deep layers perform better during the training. It is also important to consider that adding a higher number of layers means increasing the computational complexity and therefore the time required to perform the training. For this reason, for a lower number of classes where increasing the number of layers give not rise to an important increase of the quality of the training, it might be better to use a net with a lower number of deep layers, such as ResNet18.



layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times23$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 11: Different residual networks architectures. Courtesy by [13].

4 AnaSP 1.0

The first publication regarding the *AnaSP* tool was published in 2015: "*AnaSP: A software suite for automatic image analysis of multicellular spheroids*" [1]. The need for this technology comes from the knowledge that monolayer cultures are a deficient model to validate the effects of drugs and treatments for different human care application. Numerous cell types have been shown to behave differently when cultured in 3D conditions. Animal models are extensively used to perform *in vivo* experiments, but this is highly controversial both due ethical and scientific reasons. For instance, interspecies' differences in drug metabolism and toxicity between human and animal cells can jeopardize the assessment of the efficacy of the treatments. For this reason, more and more laboratories produce and use 3D cultures as *in vivo* models of tumours, spanning the gap between 2D cultures and anima models. The *AnaSP* software was proposed as a tool for analysing spheroids and automatically computing different morphological parameters, called features, that can be correlated with the effects of drugs dosage and treatments. There are many attractive to this software: first, it has been designed for experiments not necessarily performed with an automated microscope. In particular, the image of spheroids can be acquired manually with a standard widefield microscope. This software is freely available as an open-source tool written in MATLAB both as source code and standalone application. It is designed inside the framework of a user-friendly GUI (Graphical User Interface) that does not require no programming skills (Figure 12). The GUI was designed by using the MATLAB application “*AppDesigner*”, that gives the possibility to manually create an interface to be coupled with the relative code.

It is separated in two sections: the first containing the tools to perform the segmentation of the spheroids, and the second created to perform the data estimation and parameters extraction. Each button inside the GUI is coupled with a help menu, making the app particularly easy to use.

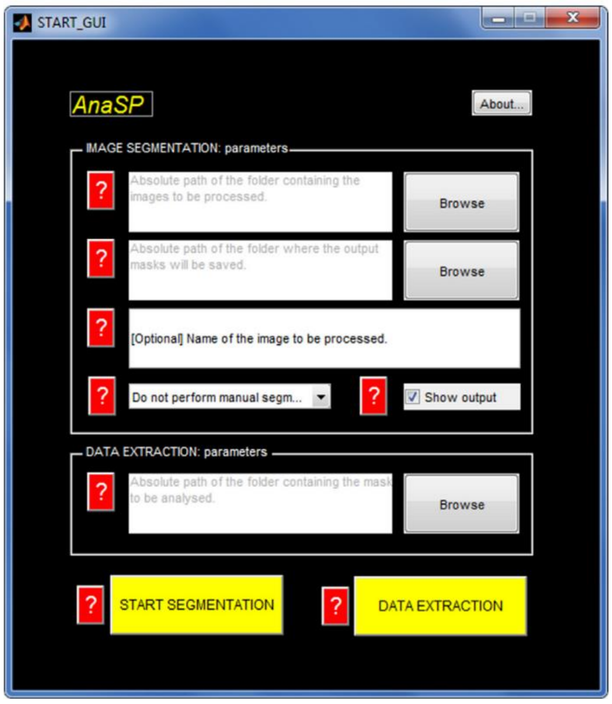

START_GUI
AnaSP
About...
IMAGE SEGMENTATION: parameters
Absolute path of the folder containing the images to be processed.
Browse
Absolute path of the folder where the output masks will be saved.
Browse
[Optional] Name of the image to be processed.
Do not perform manual segm...
Show output
DATA EXTRACTION: parameters
Absolute path of the folder containing the mask to be analysed.
Browse
START SEGMENTATION
DATA EXTRACTION

Figure 12: AnaSP 1.0 GUI snapshot. Courtesy by [1].

4.1 Image segmentation

The first module is the one responsible for segmenting the images. The only required inputs are:

- the absolute folder path of the folder containing the images the user wants to segment;
- the absolute folder path of the folder in which the output segmented images have to be saved;
- the name of the particular image the user wants to segment, in the situation were only one of the folder images needs to be segmented. This is an optional field that have to be left untouched if the segmentation has to be performed on the whole folder;
- the segmentation method, that can be chosen by the user through a pull-down list;
- the *"show output"* button, that shows a preview of every segmented mask alongside the input image.

One of the main attractive of this software is that it is designed to work with brightfield images simply acquired using common widefield microscopes, meaning that no expensive technologies are required. The spheroids must be well contrasted on the background and the border must be in focus. Starting from the brightfield images, *AnaSP* automatically computes the binary masks of the spheroids and the related parameters. The only assumption required is that the spheroids characterized by a locally spherical symmetry, under the requirement that each image contains a single spheroid. If multiple spheroids are present in the same input image the software will automatically analyse the biggest one. The spheroids images used as input must be well contrasted on the background and the border of the spheroids must be in focus. The images are first automatically converted in grey levels. If the background is unevenly illuminated, correction is suggested to visualize the spheroid as a homogeneous black mass.

The segmentation process is the core of *AnaSP*. Once the input image has been correctly pre-processed, it is segmented by following a coarse-to-fine approach based on the analysis of the intensity histogram. This histogram is analysed in order to find bimodal distribution. In fact, if the images are correctly pre-processed, all the spheroid pixels and the background pixels, will be distributed around a very different grey level intensity. This means that we can automatically define the optimal threshold that is able to divide the spheroids and background distribution. A second fine segmentation is then performed similarly to what is explained in the article [15]. An example of the segmentation procedure is provided in Figure 13, where the segmentation was performed on an image of calcium-phosphate granules. The thresholding algorithm used was the famous Otsu thresholding, that search for the threshold that minimizes the intra-class variance, defined as the weighted sum of the variances of the two classes:

$$\sigma_w^2(t) = \omega_0(t)\sigma_0^2(t) + \omega_1(t)\sigma_1^2(t)$$

Where ω_0 and ω_1 are the probabilities of the two classes separated by the threshold “t” and σ_0 and σ_1 are the variances of these two classes. Minimizing this value is equivalent to maximizing the inter-class variance, calculated as:

$$\sigma_b^2(t) = \omega_0(t)\omega_1(t)[\mu_0(t) - \mu_1(t)]^2$$

Where μ_1 and μ_0 are the means over the two different classes, and are calculated as:

$$\mu_0 = \frac{\sum_{i=0}^{t-1} ip(i)}{\omega_0(t)}; \mu_1 = \frac{\sum_{i=t}^{L-1} ip(i)}{\omega_1(t)}$$

Where L is the total number of bins in the histogram and $p(i)$ are the probabilities that each bin corresponds to the relative class.

Then, after the thresholding operations, some morphological operations are computed in order to improve the segmentation. Specifically, a closing followed by hole filling is performed. Finally, a size area filtering is performed, and all the object apart from the bigger one detected are deleted. They are in fact often representation of dust or smaller aggregates that are useless for the aim of the segmentation.

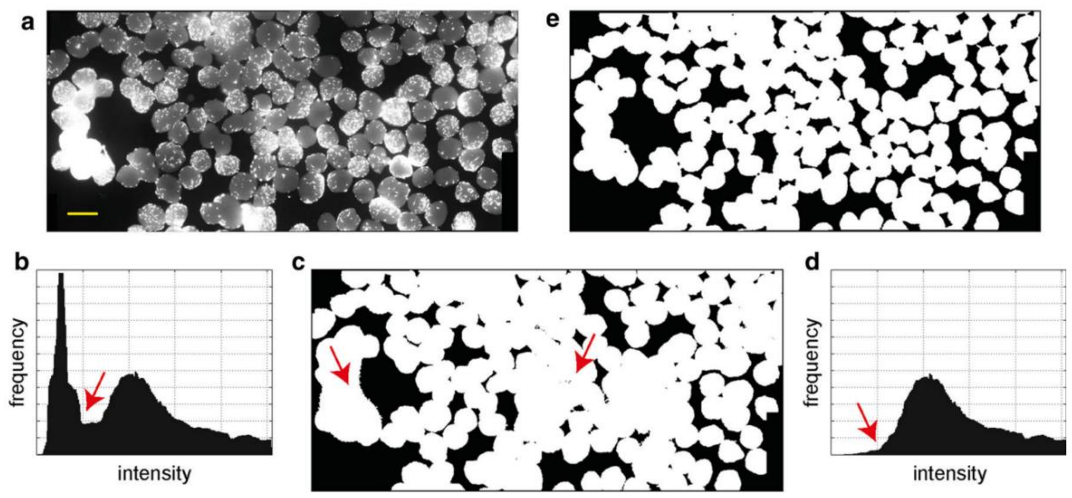


Figure 13: a) Original picture. b) Intensity histogram. The red arrow points out the first valley used as threshold. c) Raw mask obtained from the thresholding. d) Intensity histogram of the obtained mask. The red arrow points the new threshold. e) Final mask.

Apart from the automatic threshold-based segmentation, in order to be able to cope also with cases where that fails, two additional modules were implemented in *AnaSP* 1.0 to allow the user to segment all the images acquired. In fact, the presence of non-homogeneous background can lead to errors in the analysis of the intensity histogram, meaning that segmentation errors can occur. In these cases, the software asks the user to manually select a smaller region of interest (ROI) containing the spheroid. This ROI is then analysed by following the previous described steps leading to a successful segmentation most of the times. Finally, in those cases in which also the semi-automatic segmentation fails, an interactive module for spheroid segmentation is proposed to allow the user to manually segment the spheroid by drawing a line following the border.

A visual example of the complete segmentation procedure with the final refined binary mask is proposed in Figure 14.

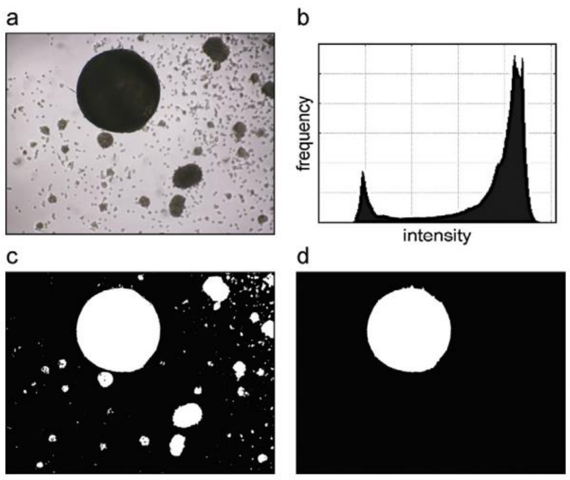


Figure 14: Example of the segmentation step performed by AnaSP: (a) input spheroid image; (b) intensity histogram; (c) mask obtained with the threshold method; (d) cleaned final binary mask. Courtesy by [1].

4.2 Data extraction

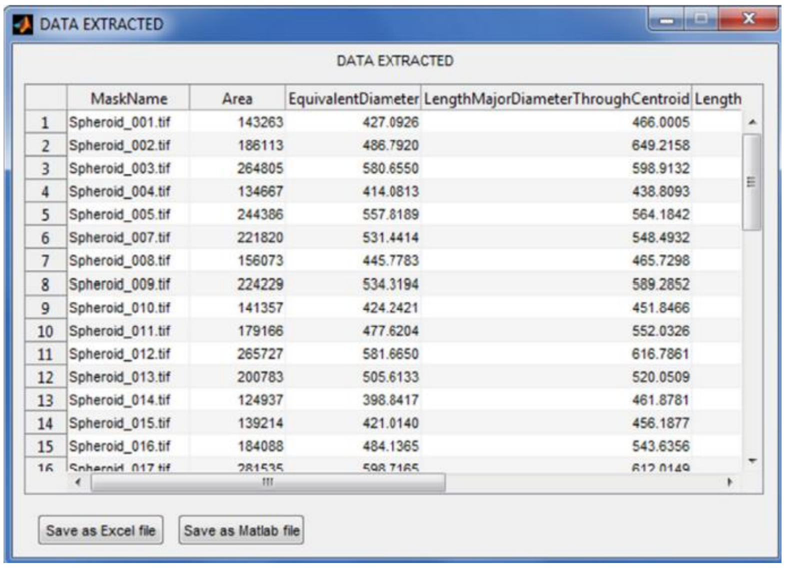
Finally, the last module of *AnaSP* 1.0 is the feature extraction module. The architecture is modular, meaning that the user is able to select which morphological feature will be compute for each spheroid, with the possibility of adding new parameters to be computed. Here the user is able to select the morphological parameters to be automatically computed for each spheroid. The default included and estimated parameters in this module are:

- Major and minor axes: is the length in pixels of the maximum and minimum straight lines that pass through the centre of the spheroid;

- Diameter of the equivalent circle: is the diameter of the circle that have the same area of the cross-section of the analysed spheroid;
- Perimeter (P): is the number of pixels composing the outer border of the spheroid;
- Area (A): is the number of pixels belonging to the cross-section of the spheroid in the analysed mask;
- Volume (V): is the 3D rendering obtained from the 2D spheroid cross section generated by supposing a spherical symmetry around the maximum axis. It is finally computed as the number of voxels contained in the obtained 3D rendering.
- Sphericity (S): is the value computed according to the following equation.

$$\mathrm{S} = \frac{\pi\sqrt{\frac{4A}{\pi}}}{P}$$

These estimates are then stored in a table that reports for each row the name of the original image and in columns the values of the parameters, as is presented in Figure 15. This table can be exported both as a Microsoft Excel table or as a MATLAB file.



DATA EXTRACTED

	MaskName	Area	EquivalentDiameter	LengthMajorDiameterThroughCentroid	Length
1	Spheroid_001.tif	143263	427.0926	466.0005	
2	Spheroid_002.tif	186113	486.7920	649.2158	
3	Spheroid_003.tif	264805	580.6550	598.9132	
4	Spheroid_004.tif	134667	414.0813	438.8093	
5	Spheroid_005.tif	244386	557.8189	564.1842	
6	Spheroid_007.tif	221820	531.4414	548.4932	
7	Spheroid_008.tif	156073	445.7783	465.7298	
8	Spheroid_009.tif	224229	534.3194	589.2852	
9	Spheroid_010.tif	141357	424.2421	451.8466	
10	Spheroid_011.tif	179166	477.6204	552.0326	
11	Spheroid_012.tif	265727	581.6650	616.7861	
12	Spheroid_013.tif	200783	505.6133	520.0509	
13	Spheroid_014.tif	124937	398.8417	461.8781	
14	Spheroid_015.tif	139214	421.0140	456.1877	
15	Spheroid_016.tif	184088	484.1365	543.6356	
16	Spheroid_017.tif	281535	598.7165	612.0149	

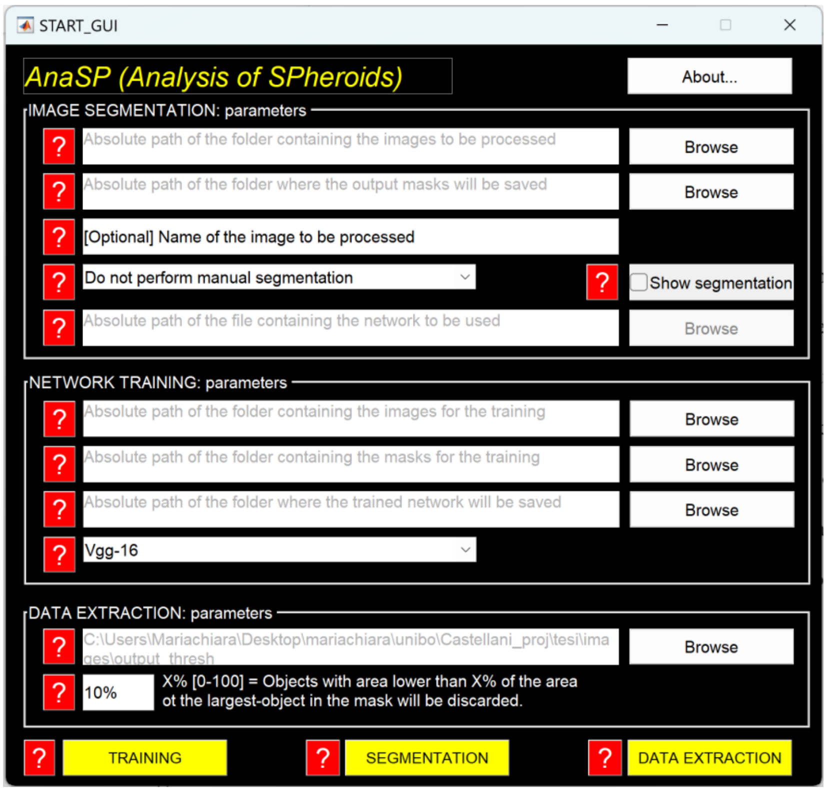
Figure 15: snapshot of a table of extracted features from AnaSP 1.0. Picture courtesy by [1].

5 AnaSP 2.0

Starting from the first version of *AnaSP*, we implemented a new module for automatically segmenting 2D brightfield images of spheroids by exploiting convolutional neural networks. Several deep learning segmentation models (*i.e.* VVG16, VGG19, ResNet18, ResNet50) have been trained and compared. All of them obtained very interesting results and ResNet18 ranked as the best-performing [16].

All the old functionalities in *AnaSP* 1.0 have been maintained, and most of them have been extended. In particular, the input images can now be both grey-level and RGB, and they can be uploaded in different formats, such as tif, tiff, png, bmp, and jpg with 8, 12 or 16 bits.

AnaSP 2.0 has been developed in MATLAB and is freely available both as source code and as standalone application. The new standalone version is presented with an improved GUI that contains all the new functionalities, without compromising the ease of use of the application (Figure 16). The new release is compatible with Windows, Macintosh, and UNIX-based systems and is already integrated with an 18-layer deep residual network trained for spheroid segmentation. The source code, the compiled standalone versions for Windows, Mac, and Linux, a user manual, sample images, a video tutorial, and further documentation are freely available at: *https://sourceforge.net/p/anasp*.


START_GUI

AnaSP (Analysis of SPheroids)

IMAGE SEGMENTATION: parameters

Absolute path of the folder containing the images to be processed

Absolute path of the folder where the output masks will be saved

[Optional] Name of the image to be processed

Do not perform manual segmentation

Show segmentation

Absolute path of the file containing the network to be used

NETWORK TRAINING: parameters

Absolute path of the folder containing the images for the training

Absolute path of the folder containing the masks for the training

Absolute path of the folder where the trained network will be saved

Vgg-16

DATA EXTRACTION: parameters

C:\Users\Mariachiara\Desktop\mariachiara\unibo\Castellani_proj\tesi\images\output_thresh

10%

X% [0-100] = Objects with area lower than X% of the area ot the largest-object in the mask will be discarded.

Figure 16: snapshot of the AnaSP 2.0 improved GUI.

5.1 Segmentation module

The segmentation module has been heavily improved with respect to the *AnaSP* 1.0 version. The interface for this module, presented in Figure 17, is composed by six field, five of which serves the same purpose they have in the previous release.

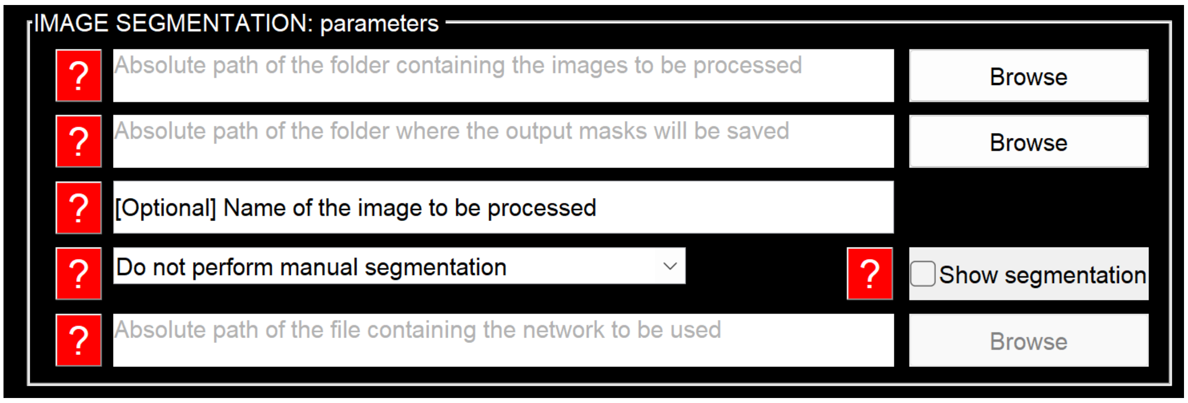


IMAGE SEGMENTATION: parameters

Absolute path of the folder containing the images to be processed

Absolute path of the folder where the output masks will be saved

[Optional] Name of the image to be processed

Do not perform manual segmentation

Show segmentation

Absolute path of the file containing the network to be used

Figure 17: Segmentation module AnaSP 2.0

They are: the field were the images to be segmented will be inserted; the field where it has to be selected the folder in which the segmented images will be saved; the optional field where the user can enter the precise name of the image to be segmented in the eventuality in which it is necessary to segment only one image instead of the entire folder; and the tick *"show segmentation"* that gives to the user the possibility of seeing each segmented image alongside its referred original image. In particular, if this option is selected, after each image inside the selected folder is segmented, the user will be presented with an image shown in Figure 18. The algorithm will pause until the user presses a random key, then the figure will clear itself and the operation will proceed as usual.

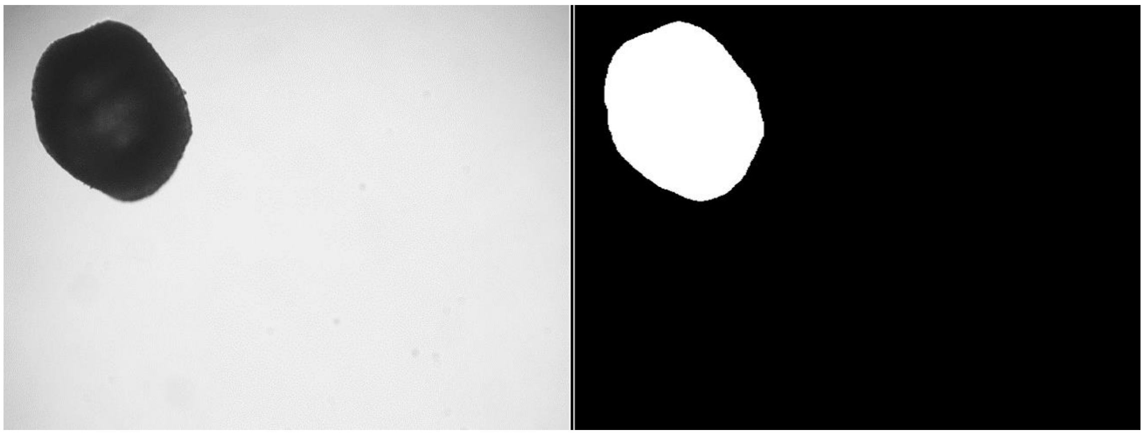


Figure 18: example of the plot shown in AnaSP 2.0 if the option "show segmentation" is selected.

All the old segmentation methods, such as the automatic threshold-based segmentation and the manual segmentation, are still present. Along with them, a new option has been introduced: the automatic segmentation performed using a pre-trained neural network. As for the *AnaSP* 1.0 release, in order to choose between all the methods, the user needs to select the correct one from the drop-down menu shown in Figure 19.

The sixth field in the segmentation module (Figure 17), concerning the absolute path of the file containing the network to be used for the segmentation, have to be used only if the new method realized in this new release of the software is selected from the drop-down menu.

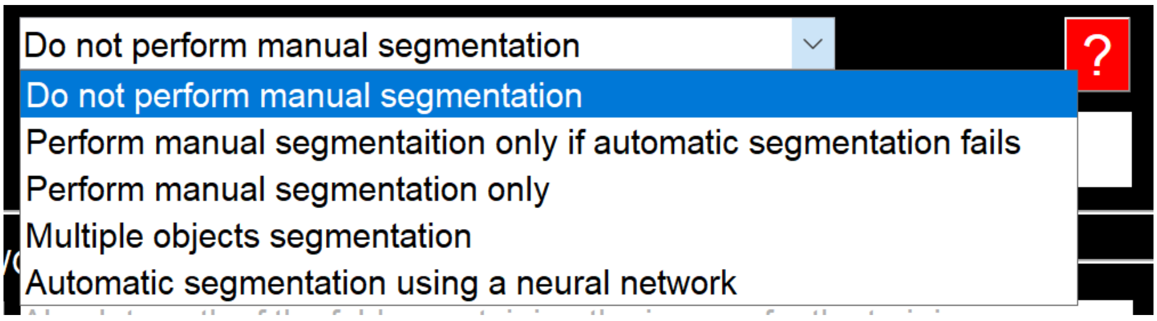
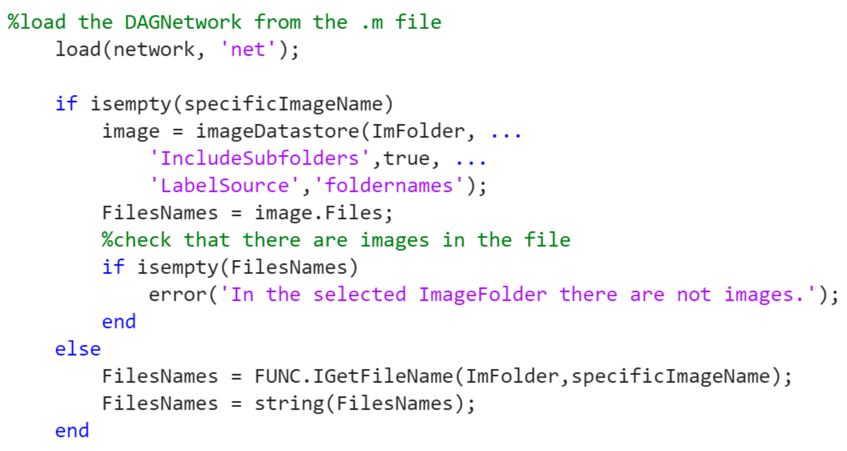

Do not perform manual segmentation
?
Do not perform manual segmentation
Perform manual segmentaition only if automatic segmentation fails
Perform manual segmentation only
Multiple objects segmentation
Automatic segmentation using a neural network

Figure 19: Drop-down menu of the segmentation module in the AnaSP 2.0 GUI.

In particular, if the *"Automatic segmentation using a neural network"* is selected, the user will be asked to also select the pre-trained network to be used to perform the segmentation. This will be done by importing in the correct field the ".mat" file containing the correct DAGNetwork variable. An already trained network based on ResNet18, ready to use for spheroid segmentation, is integrated inside the files of the open-source code uploaded on the website.

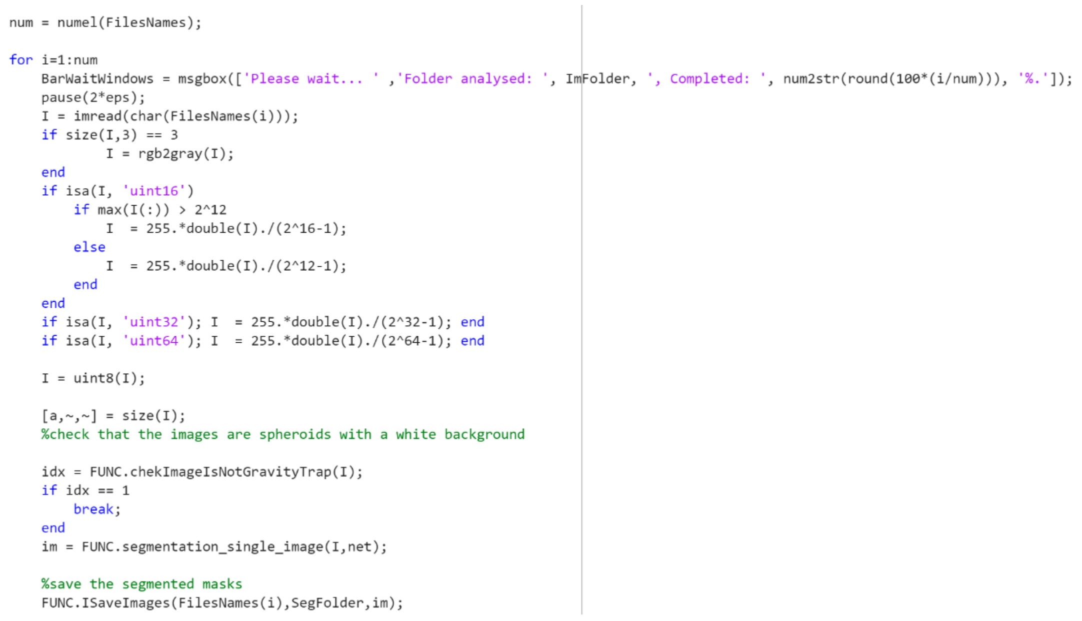
The segmentation is performed through a specifically designed function called "*segmentation_multiple_images*". In the first part of the code, reported in Figure 20, the segmentation network is loaded. Then the code creates the variable containing the images to be segmented. In particular, the variable will contain all the images contained into the folder if the "*specificImageName*" variable is empty, and only the specific selected images if the user inserts in the optional field the name of a particular image.



```
%load the DAGNetwork from the .m file
    load(network, 'net');

    if isempty(specificImageName)
        image = imageDatastore(ImFolder, ...
            'IncludeSubfolders',true, ...
            'LabelSource','foldernames');
        FilesNames = image.Files;
        %check that there are images in the file
        if isempty(FilesNames)
            error('In the selected ImageFolder there are not images.');
        end
    else
        FilesNames = FUNC.IGetFileName(ImFolder,specificImageName);
        FilesNames = string(FilesNames);
    end
```

Figure 20: Input images upload in the segmentation_multiple_images function

Then, each of the images contained in the "FileNames" variables is analysed and segmented by the code presented in Figure 21. In particular, each image is converted into a grey level image and, if it is of type "uint16", "uint32" or "uint64" it is processed to be of "uint8". This is due to the fact that the segmentation is built to work the best with "uint8" images. Then the code checks that the input image was not acquired through gravity trap, because the automatic segmentation was not trained for this kind of images. Finally the single image is segmented and then saved into the output folder selected by the user in the GUI.

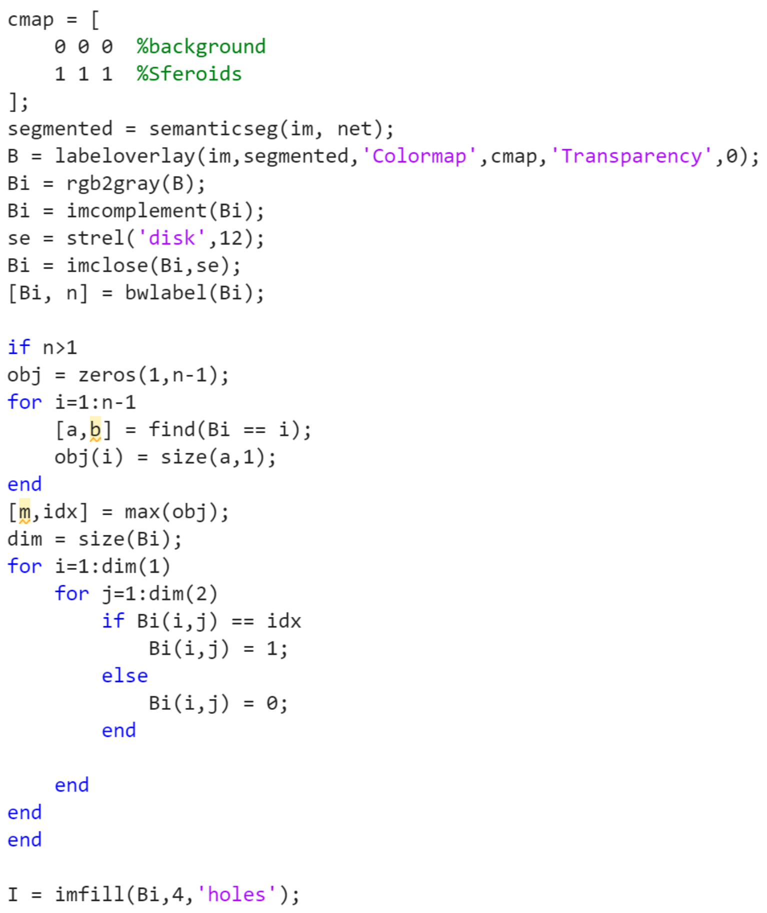


```
num = numel(FilesNames);

for i=1:num
    BarWaitWindows = msgbox(['Please wait... ' ,'Folder analysed: ', ImFolder, ', Completed: ', num2str(round(100*(i/num))), '%.']);
    pause(2*eps);
    I = imread(char(FilesNames(i)));
    if size(I,3) == 3
            I = rgb2gray(I);
    end
    if isa(I, 'uint16')
        if max(I(:)) > 2^12
            I  = 255.*double(I)./(2^16-1);
        else
            I  = 255.*double(I)./(2^12-1);
        end
    end
    if isa(I, 'uint32'); I  = 255.*double(I)./(2^32-1); end
    if isa(I, 'uint64'); I  = 255.*double(I)./(2^64-1); end

    I = uint8(I);

    [a,~,~] = size(I);
    %check that the images are spheroids with a white background

    idx = FUNC.chekImageIsNotGravityTrap(I);
    if idx == 1
        break;
    end
    im = FUNC.segmentation_single_image(I,net);

    %save the segmented masks
    FUNC.ISaveImages(FilesNames(i),SegFolder,im);
```

Figure 21: Preprocessing and segmentation of the input images.

In particular, the segmentation is performed inside the "*segmentation_single_image"* function. Here the code first performs the segmentation with the built-in "*semanticseg*" function. Then the categorical output is overlayed over the input image to create the first approximate grey level mask. Next, the code performs a closure operation to correct eventual border discontinuities. Later, it checks how many objects there are in the mask.

If more than one object is detected, the code clears all the smaller ones, keeping only the bigger object, that is assumed to be the spheroid. Finally, eventual holes are closed through a fill hole operation.



```
cmap = [
    0 0 0  %background
    1 1 1  %Sferoids
];
segmented = semanticseg(im, net);
B = labeloverlay(im,segmented,'Colormap',cmap,'Transparency',0);
Bi = rgb2gray(B);
Bi = imcomplement(Bi);
se = strel('disk',12);
Bi = imclose(Bi,se);
[Bi, n] = bwlabel(Bi);

if n>1
obj = zeros(1,n-1);
for i=1:n-1
    [a,b] = find(Bi == i);
    obj(i) = size(a,1);
end
[m,idx] = max(obj);
dim = size(Bi);
for i=1:dim(1)
    for j=1:dim(2)
        if Bi(i,j) == idx
            Bi(i,j) = 1;
        else
            Bi(i,j) = 0;
        end

    end
end
end

I = imfill(Bi,4,'holes');
```

Figure 22: Function that performs the segmentation of the single image.

Finally, of the “show segmentation” check was selected, the segmented images are plotted for the user to see.

After all the parameters have been correctly defined, the user is able to start the segmentation process with a click on the yellow button “*SEGMENTATION*”.

5.2 Training Network module

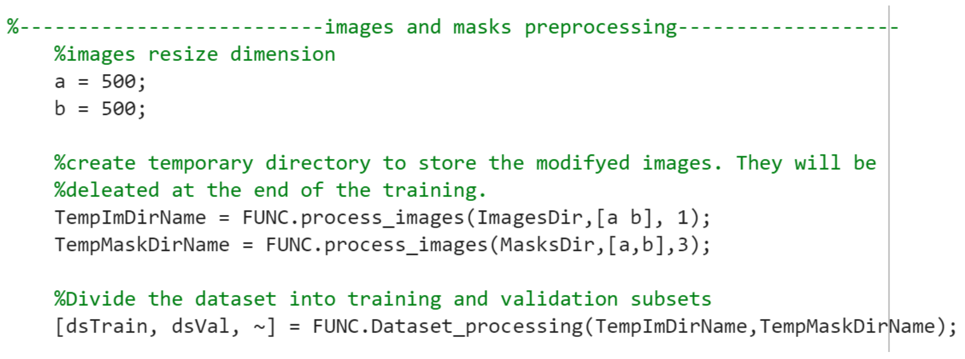
Among the new improvements added to *AnaSP* 2.0, an entirely new module has been included. This module, called *"NETWORK TRAINING: parameters",* is entirely dedicated to the training of new segmentation neural networks for spheroid segmentation.

In order to be able to use this module, the user needs to give 4 inputs:

- The absolute path of the folder containing the spheroid images that are to be used during the training;
- The absolute path of the folder containing the relative masks that are to be used during the training;
- The path of the folder in your pc where you want to save the trained network after the end of the operation. It will be saved as a .mat file named after the network you choose as your starting point associated with the time stamp representing the end of the operations, in order to have a unique identifying name for each trained network;
- The convolutional neural network you want to use as the model for the network training. It is possible to choose from the four pre-trained network enlisted before, VGG16, VGG19, ResNet18 and ResNet50, from a drop-down menu in the GUI.

The images and the masks in the input folders must be at least twenty, ten masks and ten images, for the algorithm to be able to run correctly. Furthermore, due to the fact that the input masks must be the ones obtained from the input image, the images and the masks should be the same number. In particular, for the same reason, the masks selected must have the same dimensions and names of the images and must be in the same order inside the folders to be able to correctly perform the training. It is advised, in order to obtain a reliable trained network, to use a large number of images and masks. In fact, a reduced number of images would mean a poorly trained network and consequently a poor segmentation. The training can take from few minutes to few hours to complete.

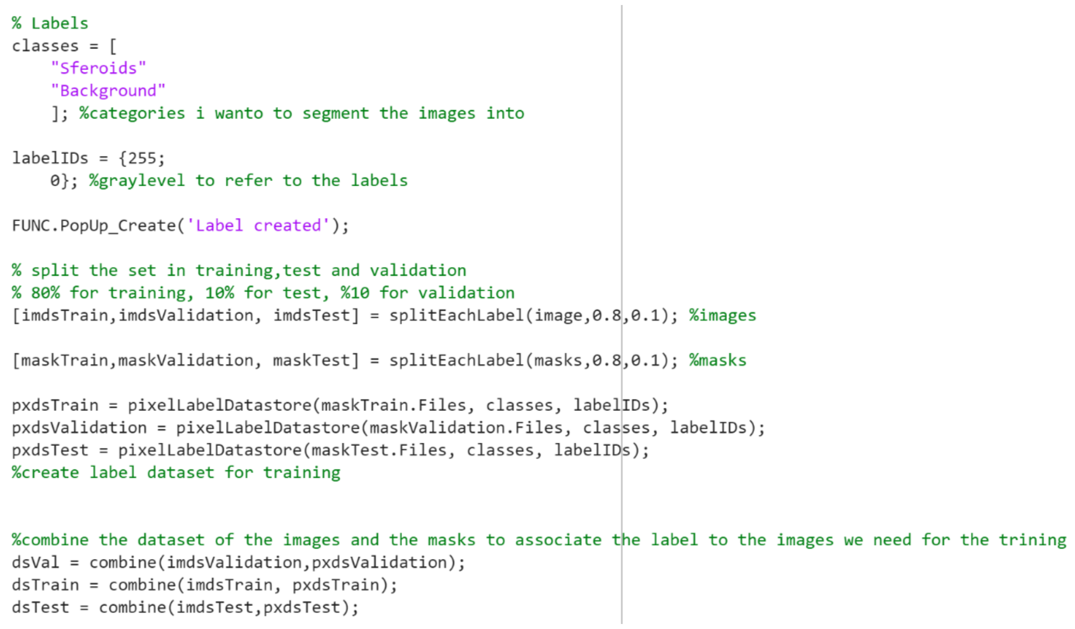
The code is divided into three main parts: image preprocessing, network selection and initialization, and network training. Firstly, the input images and masks must be pre-processed to fit into the requirements of the training process (Figure 23). This is done firstly by resizing the input images to be 500x500 greyscale images. This images are then saved into temporary directories, called "TempIm" and "TempMask" that will be automatically deleted at the end of the training process.



```
%--------------------------images and masks preprocessing-------------------
    %images resize dimension
    a = 500;
    b = 500;

    %create temporary directory to store the modifyed images. They will be
    %deleated at the end of the training.
    TempImDirName = FUNC.process_images(ImagesDir,[a b], 1);
    TempMaskDirName = FUNC.process_images(MasksDir,[a,b],3);

    %Divide the dataset into training and validation subsets
    [dsTrain, dsVal, ~] = FUNC.Dataset_processing(TempImDirName,TempMaskDirName);
```

Figure 23:Images preprocessing section of the network training function.

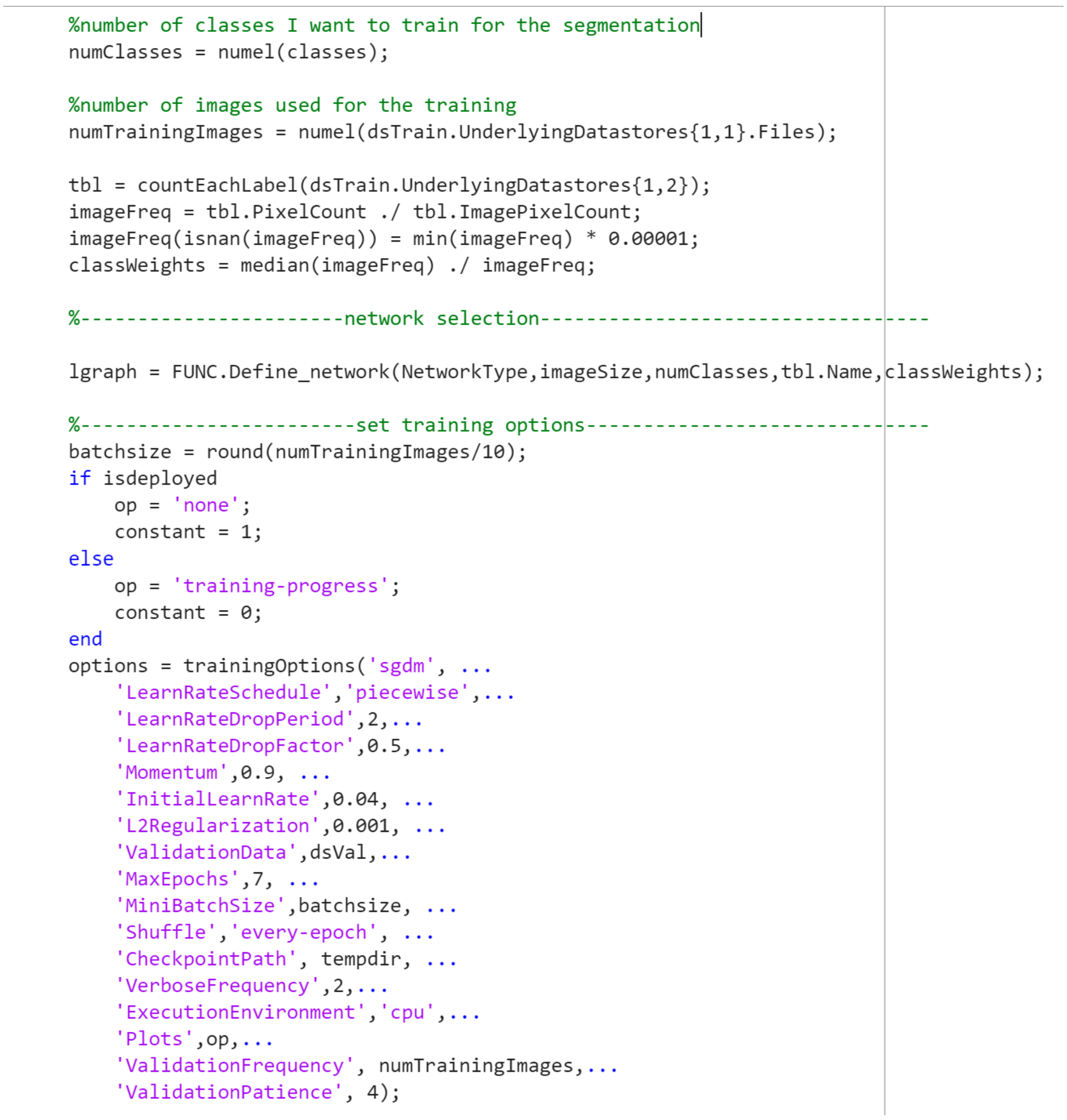
Then the resized images are processed to create the pixel label datastores that will be used for the training. To do so, the function "*Dataset_processing*" was created (Figure 24). This function takes the resized images and, after dividing the input datastores in to three groups, one for training, one for validation, and one for testing, it creates the relative pixel label datastores. This is done by associating to each pixel the correct class based on the grey level in the mask and then combining them with the respective image datastore.



```
% Labels
classes = [
    "Sferoids"
    "Background"
    ]; %categories i wanto to segment the images into

labelIDs = {255;
    0}; %graylevel to refer to the labels

FUNC.PopUp_Create('Label created');

% split the set in training,test and validation
% 80% for training, 10% for test, %10 for validation
[imdsTrain,imdsValidation, imdsTest] = splitEachLabel(image,0.8,0.1); %images

[maskTrain,maskValidation, maskTest] = splitEachLabel(masks,0.8,0.1); %masks

pxdsTrain = pixelLabelDatastore(maskTrain.Files, classes, labelIDs);
pxdsValidation = pixelLabelDatastore(maskValidation.Files, classes, labelIDs);
pxdsTest = pixelLabelDatastore(maskTest.Files, classes, labelIDs);
%create label dataset for training


%combine the dataset of the images and the masks to associate the label to the images we need for the trining
dsVal = combine(imdsValidation,pxdsValidation);
dsTrain = combine(imdsTrain, pxdsTrain);
dsTest = combine(imdsTest,pxdsTest);
```

Figure 24: Code developed to create the datastores to be used for the training.

Then the network chosen by the user is correctly initialized inside the code by using the "Define_network" function and right after the training begins (Figure 25). The options for the training are initialized based on the input datastore characteristics, in particular depending on the number of images and the frequency in which each label appears in the masks.

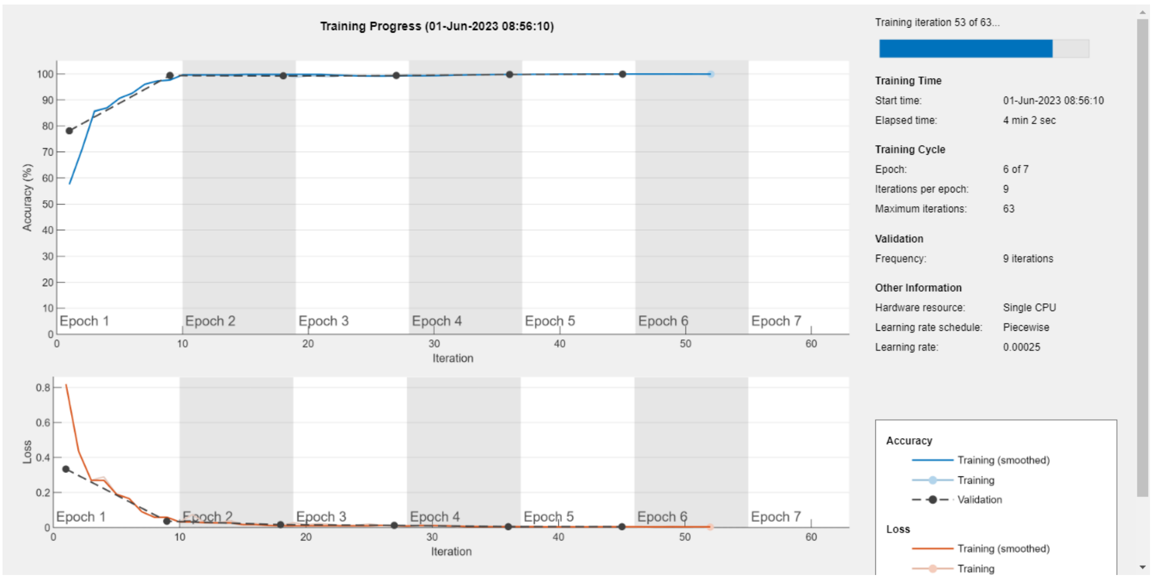


```matlab
%number of classes I want to train for the segmentation
numClasses = numel(classes);

%number of images used for the training
numTrainingImages = numel(dsTrain.UnderlyingDatastores{1,1}.Files);

tbl = countEachLabel(dsTrain.UnderlyingDatastores{1,2});
imageFreq = tbl.PixelCount ./ tbl.ImagePixelCount;
imageFreq(isnan(imageFreq)) = min(imageFreq) * 0.00001;
classWeights = median(imageFreq) ./ imageFreq;

%-----------------------network selection----------------------------------

lgraph = FUNC.Define_network(NetworkType,imageSize,numClasses,tbl.Name,classWeights);

%------------------------set training options------------------------------
batchsize = round(numTrainingImages/10);
if isdeployed
    op = 'none';
    constant = 1;
else
    op = 'training-progress';
    constant = 0;
end
options = trainingOptions('sgdm', ...
    'LearnRateSchedule','piecewise',...
    'LearnRateDropPeriod',2,...
    'LearnRateDropFactor',0.5,...
    'Momentum',0.9, ...
    'InitialLearnRate',0.04, ...
    'L2Regularization',0.001, ...
    'ValidationData',dsVal,...
    'MaxEpochs',7, ...
    'MiniBatchSize',batchsize, ...
    'Shuffle','every-epoch', ...
    'CheckpointPath', tempdir, ...
    'VerboseFrequency',2,...
    'ExecutionEnvironment','cpu',...
    'Plots',op,...
    'ValidationFrequency', numTrainingImages,...
    'ValidationPatience', 4);
```

Figure 25: Second section of the training network code.

If the training is performed by using the source code inside MATLAB, as the process proceeds, you will be able to follow the progression of the training by looking at the Training Progress plot that will automatically appear. This graph, showed in Figure 26, will show the accuracy of the training alongside the training loss, meaning the loss of information of our trained network. On the right side of the graph, the network parameters, automatically defined by the algorithm, are presented, alongside a legend.

At the end of the process, if the user is interested in the data presented in this plot, they can manually save it as a figure.



Training iteration 53 of 63...

Training Time

Start time:	01-Jun-2023 08:56:10
Elapsed time:	4 min 2 sec

Training Cycle

Epoch:	6 of 7
Iterations per epoch:	9
Maximum iterations:	63

Validation

Frequency:	9 iterations

Other Information

Hardware resource:	Single CPU
Learning rate schedule:	Piecewise
Learning rate:	0.00025

Figure 26: Example of the Training Process plot that appears during the training operation.

If, on the other hand, the training is performed through the standalone version of the software, this plot is not available due to the limitations of the MATLAB deploy tool. Instead, in order to be able to see the progression of the training without worrying about the correct flowing of the process, an indefinite progression bar will appear, ensuring that the process is continuing correctly.

Before starting the training process, the user should take into account the computational capacity of the computer on which the operation is going to be performed. In fact, if the input dataset is too big or the neural network has too many hidden layers, the training could fail due to the memory limits of the computer. If this is the case, the user should try a different network or a smaller image datastore.

5.3 Data Extraction module

This module remains substantially equal to the *AnaSP* 1.0 one. The main advancement is that the number of features that can be extracted is increased, there are now 15 of them. Apart from the already present ones, namely, the major and minor axes, the diameter of the equivalent circle, the area, the perimeter, the volume, and the sphericity, the new features are:

- circularity: $\frac{4\pi Area}{ConvexPerimeter^2}$.

  It describes how much the spheroid is close to being a circle. It ranges from 0 to 1. Closer the analysed spheroid is, closer the circularity value is to one.

- compactness: $\frac{4\pi Area}{Perimeter^2}$.

  This is calculated similarly to circularity, but instead of the convex perimeter the spheroid perimeter is used. It quantifies how tightly an object is packed or filled within its boundary. It ranges from 0 to 1, one meaning that it is very tightly packed.

- convexity: $\frac{\mathrm{ConvexPerimeter}}{\mathrm{Perimeter}}$.

  This parameter is a measure of the degree of convexity of the spheroid shape. In practice, it gives a quantitative measure of how many concavities there are inside the analysed shape. It ranges from 0 to 1, 1 meaning that the spheroid is fully convex.

- Feret diameter max: it is the maximum possible diameter measurable along a particular direction, in our case along the X-Y plane that contains the spheroid section represented in the binary mask [17].
- Feret diameter max orthogonal distance: length of the orthogonal axis of the box related to the feret diameter min [17].
- Feret diameter min: it is the minimum possible diameter measurable along a particular direction, in our case along the X-Y plane that contains the spheroid section represented in the binary mask [17]. It is important to note that the angle between this and the ferret diameter max is usually different from 90°.
- Feret aspect ratio: it is the ratio between the feret diameter min and the feret diameter max. The feret parameters are generally used to characterize the size, elongation, and orientation of the object. In particular the feret aspect ratio gives a measure of how much the object is elongated. It ranges from 0 to 1, where value closer to one means that the object has the max and min feret diameter close in length. This means that the spheroid is not particularly elongated along a specific direction.
- length major diameter through centroid: it is the maximum axis passing through the centroid of the foreground.
- length minor diameter through centroid: it is the maximum axis passing through the centroid of the foreground.
- solidity: it is the ratio between the spheroid area and the convex area.

The convex area being the area of the convex hull of the region, and the convex perimeter being the perimeter of the convex area. The convex hull region is the smallest convex region that contains the original spheroid and is greater or equal to the area of the original spheroid.

In addition, the procedure for easily defining new (or customized) features just by using a model template has been maintained. Using the source code of *AnaSP*, the user can decide to add or remove features to be extracted by simply including or removing files inside the folder named "*DATAtoBeExtracted*".

By default, the output values of the diameter and area measures are given in pixels and the volume in voxels. In order to convert them in the metric system, the user must compute the "*C*" coefficient by acquiring an image of an object of known size in the same setting and conditions of the other images. Then, the conversion coefficient can be calculated as: $\mathrm{C} = \frac{known-length-in-micrometers}{measured-lenght-in-pixels}$. Finally, the conversion can be made by multiplying the diameter parameter times "C", the area parameters by "C^2" and the volume by "C^3".

6 Metric evaluation: quantitative analysis of the experimental results

To validate the newly created training module and the segmentation qualities of the obtained networks, 4 different deep neural networks were trained: VGG16, VGG19, ResNet18 and ResNet50. The results published in this chapter have also been presented in the article: "*F. Piccinini, A. Peirsman, M. Stellato, J. Pyun, M. M. Tumedei, M. Tazzari, O. De Wever, A. Tesei, G. Martinelli and G. Castellani: "Deep Learning-Based Tool for Morphotypic Analysis of 3D Multicellular Spheroids. Journal of Mechanics in Medicine and Biology, DOI: 10.1142/S0219519423400341"* [16].

6.1 Network training

In order to obtain comparable results, the 4 different neural networks were trained by using the same image dataset. This was composed of approximately 10000 2048x2048 RGB spheroid images alongside their relative masks manually created. To reduce the computational effort of the training, all the input images were rescaled to 8-bit grey-level 500x500 pixels. Finally, due to the CNNs requirement of having as input images with three colour channels, artificial colour channels were created, meaning that the images used for the training have dimensions of 500x500x3 pixels. This process was automated by creating a MATLAB function called *"Process_images"* responsible for these operations. All the function described in this chapter can be found in the GitHub repository [18].

The images and masks processed in this way were then prepared to be used for the segmentation. In particular, the masks were internally labelled into 2 categories, background, and spheroid. This is done by using the MATLAB function *"Dataset_Processing"*. This is done by creating a colour map associated with the relative labels, *"Spheroid"* and *"Background"*. Every pixel in the masks is then associated with the label corresponding to the grey level, creating a categorical matrix. This is then superimposed to the input images in order to associate to every pixel of the original brightfield image the correct label. This last step is performed automatically by the MATLAB function *"PixelLabelDatastore"*. The entire dataset is then split into 3 subsets, called *dsTrain*, *dsVal* and *dsTest*, containing respectively, 80%, 10%, and 10% of the total images. These three sets will be used to perform the training (*dsTrain*) to validate the training process and check the correct progression of the algorithm while it is computing (*dsVal*), and to test the segmentation quality after the training (*dsTest*). It is crucial to test the quality of the network with new images that are not involved in the training process to avoid any possible bias.

All the networks have been trained considering 7 epochs, with 20 batches per epoch and an initial learning rate of 0.001 that changes according to a piecewise learning rate schedule with a period of 2 and a factor of 0.5. The training dataset is shuffled after every epoch to ensure that the training process is not biased due to repetition. All these operations have been performed by using the same code discussed in the section 5.2. Due to the different characteristics of the networks, the training time was different: VGG16 required approximately 6 hours, VGG19 10 hours, ResNet18 15 hours, and ResNet50 20 hours. However, it is worth noting that the computational time and complexity related to the usage of the different trained networks are comparable.

6.2 Quantitative analysis of the results

After the training of the four neural networks, there is the need to assess the quality of the segmentation obtained.

In particular, the aim of this comparison is to see which one of the trained networks has the best performance and to see how they compare with the manual segmentation performed through the manual segmentation option of the *AnaSP 1.0* software, considered as the ground truth. To perform this evaluation a folder containing a random sample of 100 segmented images was created. An example of the obtained segmentation is represented in Figure 27. By looking at the images only we can already see that the segmentation performed by VGG19 and ResNet50 are the one presenting the most evident issues. In order to thoroughly evaluate them we need to perform a quantitative analysis.

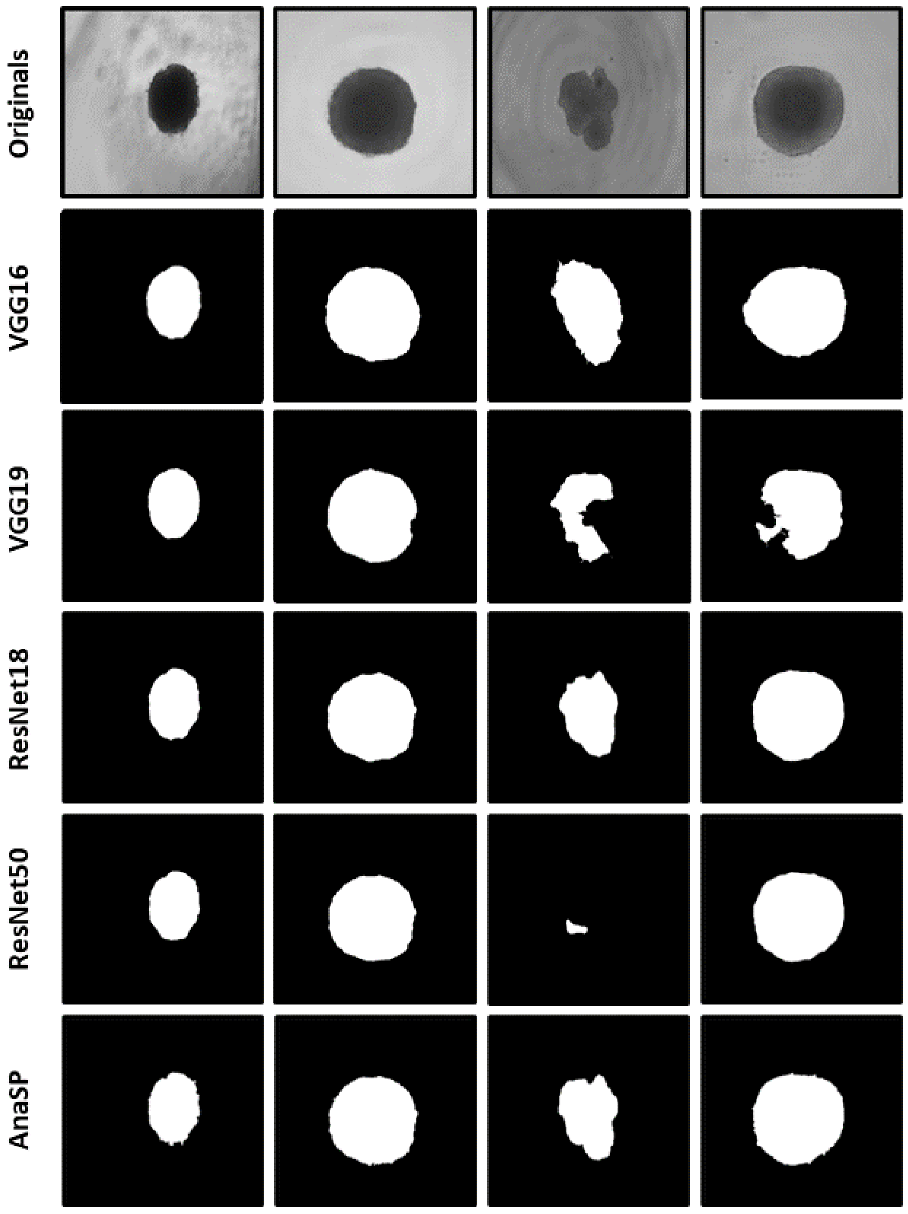


Figure 27: Top line: original images. Middle lines: masks obtained with the different trained networks. Bottom line: masks obtained with AnaSP 1.0. Images courtesy by [16].

To do this, a built-in MATLAB function, called *"evaluateSemanticSegmentation"* was used. This function takes as input the categorical dataset obtained through the segmentation and the one obtained by using *AnaSP 1.0*. this function takes as input the categorical dataset obtained through the segmentation and by using *AnaSP 1.0*, the latter considered as the ground truth. The output is a structure containing different parameters that can be used to assess the quality of the segmentation, basically calculating the degree of uncertainty we expect in general to have for the segmentation.

These parameters are:

- confusion matrix: it is presented both in absolute values and normalized over the number of pixels. It is a table that shows on the diagonal the percentage of pixels that were correctly segmented and in the other cells the percentage of misclassified pixels. In our specific case, the pixels classified as spheroids by both the automatic segmentation and the ground truth provided by *AnaSP 1.0* are called true positives (TP), the pixel classified as background in both cases are called true negatives (TN), the pixel wrongly classified as spheroids by the deep neural networks are called false positives (FP) and the pixel wrongly classified as background are called false negatives (FN).
- global accuracy (GA): it is the ratio of correctly classified pixels to the number of total pixels without regards of the class in which they were segmented. We can describe this value with the following equation: $GA = \frac{TP+TN}{TP+TN+FN+FP}$.
- mean accuracy (MA): similarly, to the global accuracy, this value gives information about the ratio between the number of correctly classified pixels and the total pixels. The difference is that now the classes are taken into consideration. In our case the accuracy of the spheroid class (SA) and the background class (BA) are calculated separately, and then the average of the results is made. In particular, we can describe this value with the following equations:

$$MA = \frac{BA+SA}{2} \text{ with } BA = \frac{TN}{TN+FP} \text{ and SA} = \frac{TP}{TP+FN}$$

- intersection over union (IoU): this parameters gives an idea of how well the new segmentation overlaps with the old one, considered as the ground truth. The intersection over union of one class is given by: $IoU = \frac{TP}{TP+FP+FN}$. As the parameter says, the intersection between the segmentation we want to assess and the ground truth is given by the TP values, while the union can be calculated as the sum of TP, FN, and FP. This concept can be better understood by looking at Figure 28, where the green circle represents the pixels segmented in that particular class in the ground truth and the yellow circle represents the one segmented in that class by our new segmentation. A good segmentation will have a perfect overlapping between the green and yellow circles, meaning that the value of the IoU is as close as possible to one. In particular, two kinds of IoU are calculated inside the structure: the mean IoU and the weighted IoU. The former beign the arithmetic mean between the IoU of alle the classes and the latter the weighted mean over the number of pixels in each class of the single IoU.

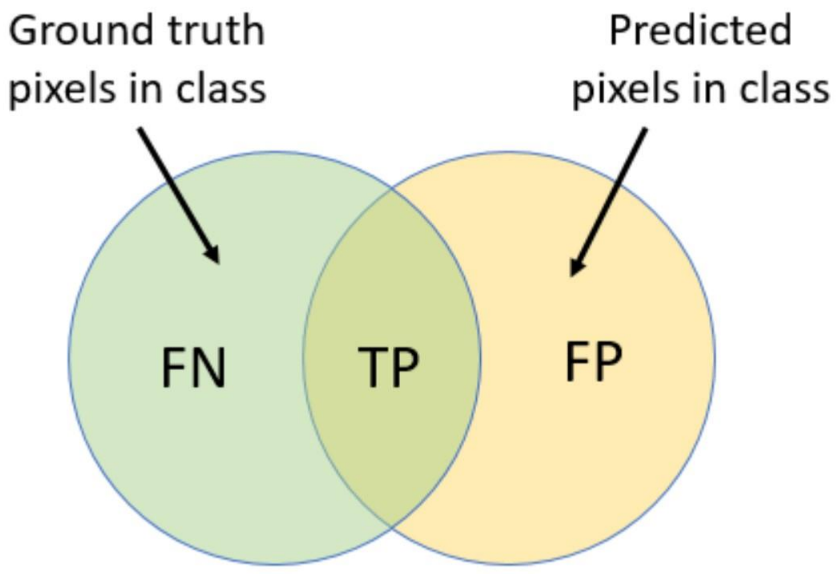


Figure 28: Intersection over union (IoU) representation. Image courtesy by [19].

- Mean BF score: also called dice coefficient, it indicates how well the predicted boundary of each class aligns with the true boundary. Practically, it is calculated as the harmonic mean between the precision measure, meaning how many between the pixels assigned to that class actually belong to that class, and the recall, meaning how many of the pixels belonging to the selected class are correctly segmented. This can be written by using the following equations:

$$Precision = \frac{TP}{TP + FP}; Recall = \frac{TP}{TP + FN}$$

Meaning that the BF score can be calculated as:

$$BF = \frac{2}{Precision^{-1} + Recall^{-1}} = 2\frac{Precision * Recall}{Precision + Recall} = \frac{2TP}{2TP + FP + FN}$$

It is calculated over each class and then the result is averaged to give the mean BF score.

More information about this tool and the structure it gives as an output can be found in the relative MathWorks pages [19] and [20].

For our purposes, in order to assess the quality of the segmentation, the normalized confusion matrix, the mean accuracy and the mean intersection over union have been computed.

The first parameter is the normalized confusion matrix, computed for all the four trained networks (Table 1). It was decided to represent the normalized confused matrix in a unique table for all the networks by identifying the matrices entrances with their meaning (*i.e.*, TP, TN, FN, FP).

Table 1: Normalized confusion matrix value for the trained networks

	VGG16	***VGG19***	***ResNet18***	***ResNet50***
TP	*0.99*	*0.99*	*0.99*	*0.99*
TN	*0.91*	*0.81*	*0.91*	*0.74*
FP	*0.09*	*0.19*	*0.09*	*0.26*
FN	*0.01*	*0.01*	*0.01*	*0.01*

Firstly, by looking at the TP values, it is evident that all the trained networks can segment the spheroid pixels in the right category with a great reliability. The problem lies mostly in the FP category, meaning that a part of the background pixels has been categorized by our segmentation as spheroid pixels. In particular, for VGG19 and ResNet50, 20-30% of the background pixels are on average misclassified. This could be expected only by looking at the segmentation example in Figure 27. On the other hand, VGG16 and ResNet18 show higher performances also for background segmentation, as less than 10% of the background pixels have been misclassified.

To further support these measurements, it is convenient to look at the mean accuracy (MA) and mean intersection over union (IoU) of each trained network. The results are reported in Table 2.

Table 2: Mean accuracy and intersection over union for the 4 trained networks.

	VGG16	***VGG19***	***ResNet18***	***ResNet50***
Mean accuracy	*0.97*	*0.93*	*0.95*	*0.87*
Mean IoU	*0.91*	*0.90*	*0.92*	*0.86*

This measures, retrieved from the normalized confusion matrices calculated for each network, can be seen as the average uncertainty over the segmentation performed. It is clear that the segmentation performed with ResNet50 is the worst performing among all the trained networks. On the other hand, the others appear by these parameters to be close to each other. Combining the information obtained by the evaluation, it resulted that VGG16 and ResNet18 were the most reliable networks. In particular, we can say that these two networks perform the training with a 97% and 95% of accuracy respectively. Obtaining better results for ResNet18 with respect to ResNet50 is something not surprising: the dataset is composed just of images with “black” spheroids on a “white” background. Especially for these cases, increasing complexity and depth (*i.e.,* number of layers) of artificial neural networks do not necessarily mean having better segmentations due to overfitting and many other problems [21].

6.3 Qualitative analysis of the results

To determine the best performing network between VGG16 and ResNet50 we can make some qualitative considerations about certain specific cases in which the segmentation fails, Figure 29 shows some representative images of these cases. It can be noted that often the segmentation failure is due to the characteristics of the input images, such as spheroids with necrotic core, where the segmentation often detects the spheroid only partially. This is due to the fact that the centre of the spheroid in these cases is the same colour of the background, making almost impossible for the network to recognize it. In general, when the spheroids have a similar colour to the background, meaning that the image contrast is not optimized, the networks show difficulties in consistently segmenting the images. Moreover, when the spheroid touches the image border, the segmentation typically fails. Finally, VGG16 in particular tends to fail if the background is not uniform.

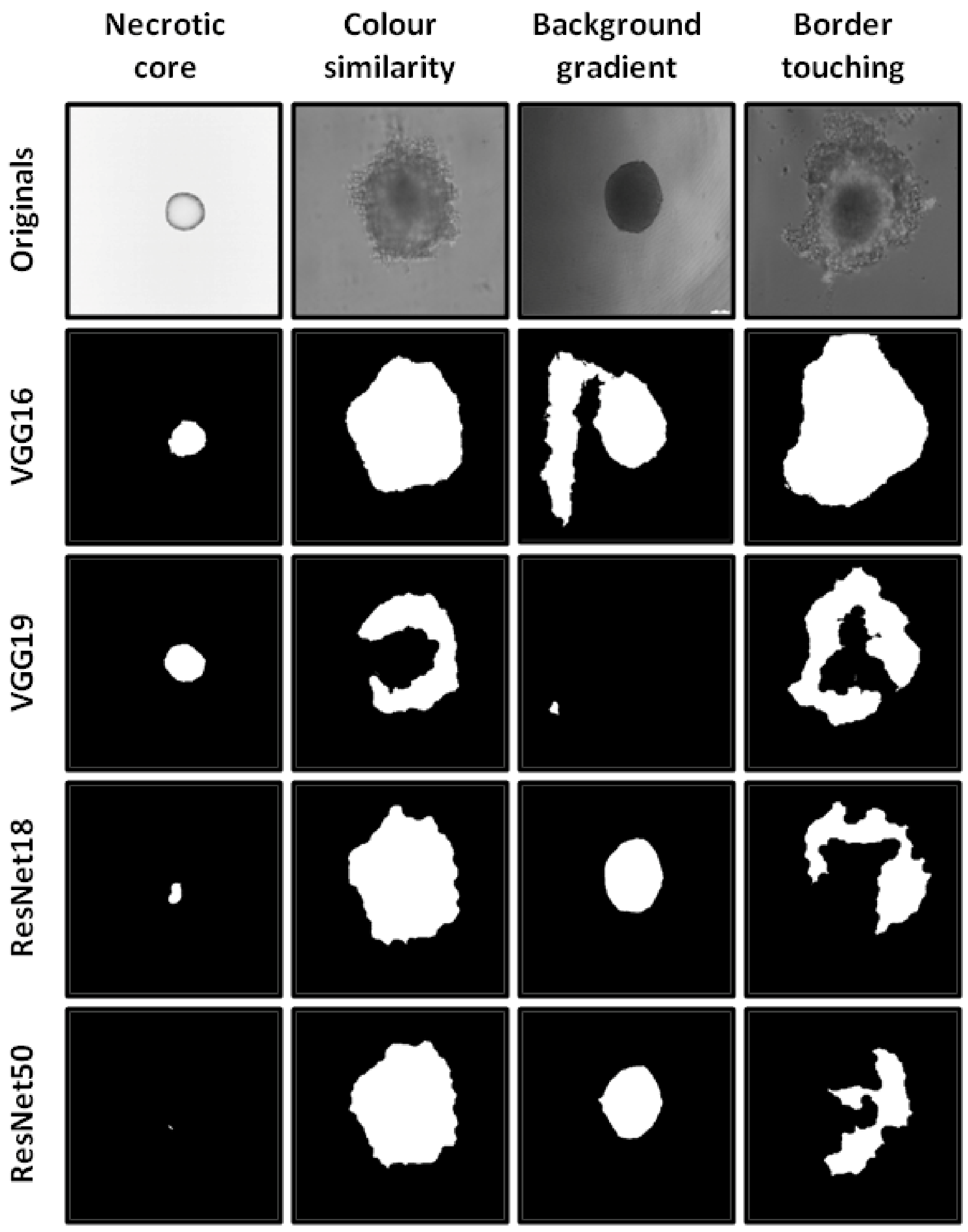


Figure 29: top line: original images representative of different segmentation problems. Other lines: masks obtained with the different trained networks. Image courtesy by [16].

Most of these issues can be easily overcome by correctly performing the input images to have the best possible contrast. The only remarkable issue is the one concerning the necrotic core spheroids, that has to be studied further.

Due to the fact that ResNet18 appears to be overall the least affected network by the background and border problems, it is deemed the one with the best overall performance.

7 Future developments

The obtained results were consistent and reliable, but there are various future developments that can be done starting from the new *AnaSP 2.0* software.

Firstly, looking at the already trained network, it is important to consider that the training was made with reduced resolution with respect to the original images' one. This means that there could be space to improve the quality of the segmentation by re-training them by using the original images. This could be possible by using a computer with higher computational complexity.

Moreover, the complexity associated with 3D *in vitro* models makes a multi-level analysis necessary, starting from the macro-observation of the drug effects on the whole spheroids to the micro analysis of the single composing cells. In fact, *in vivo* tumors are heterogeneous, and the presence of even a few "particular" cells, for instance those typically called Cancer Stem Cells (CSCs), is one of the key determinants of tumor regeneration. Identification and analysis of the CSCs are fundamental for a more accurate evaluation of drug efficacy. A standardized procedure for first analysing the effects of drugs on the whole spheroids and then performing single-cell screenings would give the opportunity to better understand the macro- and micro-behaviour of the different drugs. In order to obtain more consistent and homogeneous results, it is important to create a protocol to generate 3D spheroids with a tuned, specific shape and volume and with characteristics compliant with the requirements of a confocal microscope.

Using *AnaSP* with the new deep-learning based segmentation model can make it possible to perform preliminary experiments to characterize the morpho-biological homogeneity and stability of the spheroids generated with different cancer cell lines. There are in fact several procedures available to generate 3D spheroids, but there is no unanimous agreement regarding which method is best. In particular, there are few data regarding the pros and cons of using one technique instead of another. Practically, several systems today available to generate spheroids can be tested, and the goal will be to define a set of protocols to create, for different cancer types, groups of spheroids with a tuned shape and volume, and with characteristics compliant with the requirements of a confocal microscope (*i.e.*, dozens of homogeneous spheroids with a diameter <500 um).

Furthermore, the features extracted up to now can be mainly described as binary features, such as volume, that recover the information only from the binary segmentation masks. This means that a significant portion of the information contained in the form of grey-level pixels is lost. As a consequence, an imperative of the development of spheroid analysis is to create a new group of features, such as density, variance, entropy, *etc*., that will quantify this portion of, until now, lost information. This can be done by superimposing the categorization of every single pixel given by the segmentation with the original image. This will give the opportunity to perform various features extraction directly from the original brightfield images.

The completely developed tool can be used to analyze tumor spheroids treated with different anti-cancer drugs. The goal of this is to develop and validate an automated from macro to micro platform to perform 3D HCS of drugs using cancer multicellular spheroids. This includes a macro-analysis of the whole spheroids by extracting morphological features and quantifying the metabolites using mass spectrometry, and a micro-analysis of the single cells for performing molecular and genetic profiling on specific classes.

Lastly, we published the description of the current version of *AnaSP* in F. Piccinini, A. Peirsman, M. Stellato, J. Pyun, M. M. Tumedei, M. Tazzari, O. De Wever, A. Tesei, G. Martinelli and G. Castellani: "Deep Learning-Based Tool For Morphotypic Analysis Of 3d Multicellular Spheroids." Journal of Mechanics in Medicine and Biology, DOI: 10.1142/S0219519423400341 [16], and we would like to submit a new article describing the new version of *AnaSP*, including the above cited future works in BioInformatics-Oxford.

8 Conclusions

The main goal of this research was to create an automatized algorithm able to compute accurate segmentations of 2D brightfield images of 3D multicellular spheroids using the very fast developing technology of CNNs.

In order to do so, four deep-learning convolutional neural networks were implemented and tested: VGG16, VGG19, ResNet18 and ResNet50. Not all of them had optimal results in regards of spheroid segmentation, so the next step was to assess which one was to be used for the purpose it was created.

The obtained results have shown that the best results were given by VGG16 and ResNet18 that, with an accuracy of 97% and 95% respectively, are able to accurately segment almost all the spheroids taken into consideration. VGG19 an ResNet50 were discarded due to the fact that the segmentation was not as efficient as the two other networks, in particular the border of the spheroids was often lost in the segmentation. Going more in depth with the evaluation of the segmentation, through a qualitative analysis, it was possible to recognise some issues with the segmentation due to different artifacts and characteristics in the input images that stem in an unsuccessful final result.

The results obtained leaves space for future improvements, by specializing the training to specifically resolve those issues or by accurately pre-processing the images.

Alternatively, it can be interesting to try and train other networks that weren’t presented in this report, such as DenseNet, that requires the creation of the network from scratch, due to the fact that MATLAB does not include a tool for the modification from a classification network to a segmentation one.

This work was accepted and orally presented at the XXII International Conference on Mechanics in Medicine and Biology, 19–21 September 2022, Bologna, Italy (presentation with title: ’deep Learning Models For Segmenting Brightfield Images Of Cancer Multicellular Spheroids Used For Radiomics Analysis’ by Filippo Piccinini, Arne Peirsman, Olivier De Wever, Mariachiara Stellato, Anna Tesei, Giovanni Martinelli, Gastone Castellani. In addition, it was published as a scientific article under the name F. Piccinini, A. Peirsman, M. Stellato, J. Pyun, M. M. Tumedei, M. Tazzari, O. De Wever, A. Tesei, G. Martinelli and G. Castellani: “Deep Learning-Based Tool For Morphotypic Analysis Of 3d Multicellular Spheroids.” Journal of Mechanics in Medicine and Biology, DOI: 10.1142/S0219519423400341 [16].

Bibliography

[1] F. Piccinini, "AnaSP: a software suite for automatic image analysis of multicellular spheroids." *Computer Methods and Programs in Biomedicine,* pp. 119(1):43-52, 2015.

[2] A. Peirsman, *et al.* , "MISpheroID: a knowledgebase and transparency tool for minimum information in spheroid identity.," *Nature Methods,* pp. 18:1294-1303, 2021.

[3] T. Achilli, "Advances in the formation, use and understanding of multi-cellular spheroids," *Expert opinion on biological therapy,* pp. 1347-1360, 2012.

[4] G. B. Ware, *et al.,* "Generation of homogenous 3D pancreatic cancer cell spheroids using an improved hanging drop technique.," *Tissue Engineering Part C: Methods.,* 2016.

[5] L. K. N. Nicholas E. Timmins, "Generation of multicellular spheroids by the Haging-drop method.," *Methods in Molecular Medicine, 2nd ed.: Tissue Engineering,* 2007.

[6] Henry Page, *et al.,* "Three-dimensional tissue cultures: current trends and beyond." *Cell Tissue Res,* vol. 352, p. 123–131, 2013.

[7] Teresa Franch-Mendes, *et al.,* "Heterotypic tumor spheroids in agitation-based cultures: a scaffold-free cell model that sustains long-term survival of endothelial cells." *Frontiers in bioengineering and biotechnology,* 2021.

[8] Glauco R. Souza, *et al.,* "Three-dimensional tissue culture based on magnetic cell levitation." *nature nanotechnology,* 2010.

[9] M. E. Mayerhoefer, "Introduction to radiomics." *The journal of nuclear medicine,* pp. Vol.61-4., 2020.

[10] C. Janiesch, *et al.,* "Machine learning and deep learning." *Electronic Markets,* pp. 31:685-695, 2021, April.

[11] R. N. Keiron O'Shea, "An Introduction To Convolutional Neural Network." December 2015.

[12] L. A. , *et al.*, "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions." *Journal of Big Data,* p. 8:53, 2021.

[13] Shaoqing Ren, *et al.,* "Deep residual learning for image recognition." Dec 2015.

[14] A. Z. Karen Simonyan, "Very deep convolutional networks for large-scale image recognition." in *Published as a conference paper at ICLR 2015.*, 2015.

[15] F. Piccinini, *et al.,* "Semi-quantitative monitoring of confluence of adherent mesenchymal stromal cells on calcium-phosphate granules by using widefield microscopy images." *J. Mater. Sci. Mater. Med.,* p. 25 (10) (2014) 2395–2410, 2014.

[16] F. Piccinini, *et al.,* "Deep learning-based tool for morphotypic analysis of 3D multicellular spheroids." *Journal of Mechanics in Medicine and Biology,* 2023.

[17] MATLAB, "feret properties wrapping up." [Online]. Available: https://blogs.mathworks.com/steve/2018/04/17/feret-properties-wrapping-up/.

[18] M. Stellato, "SpheroidSegmentation_with_CNNs." [Online]. Available: https://github.com/MariachiaraStellato/SpheroidSegmentation_With_CNNs.

[19] MathWorks, "semanticSemgnetationMetrics." [Online]. Available: https://it.mathworks.com/help/vision/ref/semanticsegmentationmetrics.html.

[20] MathWorks, "evaluateSemanticSegmentation." [Online]. Available: https://it.mathworks.com/help/vision/ref/evaluatesemanticsegmentation.html.

[21] Bressen KK, *et al.,* "Comparing different deep learning architectures for classification of chest radiographs." *Scientific Reportss,* Vols. 10(1), 13590, 2020.

Acknowledgement

Having reached this stage of my journey, I think it is only right to thank the people without whom this would have not been possible. Firstly, I would like to express my deepest gratitude to Professor Gastone Castellani, Professor Filippo Piccinini and Professor Giovanni Martinelli, who gave me the opportunity of working on this innovative project, giving me also the chance to publish my first paper with them. A special mention to Professor Piccinini, that has mentored me every step of the way that led me to this milestone.

Then I would like to express my gratitude to my parents and my grandparents, who supported me through every stage of my life without exceptions and without whom this would have never been possible. I am proud to be able to have them with me in this journey. A special mention to Fra, that have supported me in these stressful months without hesitation and with whom I hope to spend many more.

I must finally say thank you to all my dear friends, with a special mention to Silvia and Martina, who walked with me through these years with unwavering support. A special thankyou to my flatmates, Giada and Alessia, with whom I shared every moment of the day for the last two years, through high and low. Without all the wonderful people that are a part of my life this moment would have remained only a dream, and for this I am thankful.