POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

CellHunter: a Deep Learning-based Model for Automatic Bone Stromal Cell Segmentation in Prostate Cancer Bone Metastasis

Tesi di Laurea Magistrale in Ingegneria Biomedica - Biomedical Engineering

Giorgio Allegri, 10806399
Luca Pavirani, 10805553

Advisor:
Prof. Pietro Cerveri

Co-advisors:
Prof. Stefano Casarin
Prof. Eleonora Dondossola

Academic year:
2022-2023

Abstract: Prostate Cancer (PCa) is a common (12.5%) and deadly disease among men, with metastases being the leading cause of mortality. Bone metastases can cause severe health issues and drastically decrease the 5-year survival rate from 100% primary to 30% metastatic tumor. In this context, pre-clinical research strongly relies on microscopy image analysis, a task that is typically performed manually by biologists. The automation of this procedure is becoming increasingly important due to its time-consuming nature, low reproducibility, and exposure to human error. We hypothesize that applying deep learning tools to microscopy image analysis will overcome the current image processing limitations and optimize pre-clinical research. Accordingly, we developed a neural network-based architecture to perform semantic segmentation of cellular structures of interest such as osteoblasts, osteoclasts, bone, and vessels from PCa bone metastasis microscopy images, followed by relevant parameter extraction, such as the number of cells, their location, and their distance. We achieve satisfying metrics, despite a rather limited data availability, in terms of Intersection over Union (IoU). OC model reached 0.73 and 0.57 in terms of training and validation IoU, respectively. OB model achieved 0.84 in training and 0.55 in validation. The bone models resulted in comparable IoU training values (0.79 MPM and 0.81 CM) and validation values (0.72 MPM and 0.71 CM). Finally, vessel model reached 0.64 in training and 0.42 in validation. Biologists tested our models, confirming the biological importance and the segmentation accuracy provided by our system. With this model, we provided biologists with a fully automated, yet reliable tool, to reduce the time required for image analysis from days to seconds, and optimize the image processing pipeline. Additionally, an executable software, CellHunter, equipped with a GUI was developed to increase usability. Future developments will focus on expanding data variability, increasing accuracy in labelling, and making the software available on other operative systems.

Key-words: Bone Metastasis, Prostate Cancer, Stromal Cells, Deep Learning, Convolutional Neural Network, U-Net, Image2patch, Semantic Segmentation, GUI, Image Analysis

1. Introduction

Prostate Cancer (PCa) is a highly fatal pathology, being one of the leading causes of cancer-related deaths in men. Its incidence is significant, affecting 1 in 8 individuals, making it the second most common cancer in men, after lung cancer [1]. The major cause of PCa-related deaths is the insurgence of metastases, that occur in over 80% of cases of advanced-stage cancer [2]. Skeletal metastases cause critical health problems, including pathological fractures, severe and constant pain, and, eventually, death. In particular, bone metastasis make the 5-year survival rate drop from nearly 100% as primary tumor, to just 30% when metastatic tumor [3, 4]. PCa cell bone invasion includes different stages that ultimately lead to bone remodelling deregulation, altering bone structure. Cells involved in bone remodeling are mainly osteoclasts (OCs) and osteoblasts (OBs). OCs are responsible for the resorption of the bone, while OBs produce and deposit new bone. The activity of these cells is tightly regulated, and for each portion of bone undergoing resorption, an equivalent amount of new bone is produced [5] to renovate bone structure while maintaining homeostasis. Tumor cells alter bone remodeling by interfering with OBs' and OCs' activity. There are two types of bone metastases:

- *Osteoblastic Bone Metastasis*: over-production of bone due to up-regulation- of OBs (and down-regulation of OCs) resulting in irregular bone structures and pain [6, 7].
- *Osteolytic Bone Metastasis*: up-regulation of OCs causing deregulated bone resorption that leads to pathological bone fractures [7].

Several anti-bone metastasis treatments are today available. Bone-targeted agents, such as zoledronic acid and denosumab, are commonly used to reduce the incidence of skeletal-related events [8, 9]. Hormone therapy, such as androgen deprivation therapy, is often used to treat metastatic prostate cancer and has been shown to reduce the risk of bone metastases [10]. Chemotherapy [11, 12], radiation therapy [13, 14], and immunotherapy [15] have also been shown to improve outcomes in patients with bone metastases. However, these agents showed promising results only on the short-term, with relapse and tumor re-growth on the long-term, making them merely palliative [4]. To improve the outcome of these therapies, it is crucial to gain a deeper understanding of how bone metastasis responds and resists to therapies.

Classical analysis of bone metastasis is done by molecular approaches, such as gene expression analysis and proteomic analysis, combined with histological analysis of macroscopic images. However, these approaches lack sensitivity in studying the interactions between tumor cells, bone stroma, and their responses to therapies [16, 17]. Microscopy image analysis provides quantitative support for improving bone metastases analysis hence, it is the most effective approach to address the problem. Advancements in imaging technology enabled visualization and tracking of the interactions between tumor cells and bone stroma, and a deep tracking of response to therapies in a 3D and time-resolved manner. As a result, these techniques provide a more comprehensive and detailed understanding of the complex interactions occurring within the bone metastasis microenvironment [16]. Due to the massive growth of image data, both in number and complexity, standard manual processing is becoming inefficient and, in many cases, unfeasible. The extraction of crucial information from these images is both time-consuming and complex, it requires specific training for the operator, and it is affected by human error. Therefore, there is a clear need for further refinement of these techniques to reduce the time required for analysis and to minimize human error. Computer methods gained significant attention in recent literature, as they offer efficiency and objectiveness in image processing [18].

Deep learning (DL) algorithms, specifically, improved image processing accuracy and reduced the time required to analyze microscopy images by automatizing the process of identification, recognition, extraction and quantification of patterns and parameters of interest [18, 19]. DL models have great flexibility, and accordingly they are employed in multiple applications such as nuclei detection [20], cell and tissue segmentation [21], medical image analysis and lesion detection [22].

At best of our knowledge, there has been no prior research on the segmentation of bone stromal cells from microscopy images. Therefore, we hypothesize that a computational algorithm can automatically segment and quantify microscopy images, significantly reducing the time required to study the bone microenvironment. Accordingly, we propose a DL-based infrastructure to optimize time and to improve the accuracy of microscopy bone image analysis employed to better understand the response of bone stroma and tumor cells to therapies. We developed the pipeline end-to-end involving: manual label segmentation, data-cleaning and pre-processing, model fitting and evaluation, and post-processing to retrieve the information of interest. Additionally, we created "CellHunter", an executable software that includes a user-friendly Graphic User Interface (GUI) that integrates our DL-based model. This software is designed to be accessible by biologists who may have limited programming knowledge, offering an easy-to-use tool for cell segmentation and post-processing. Our research is focused on the segmentation of specific bone stromal cells, including OBs, OCs, vessels, and bone profile. To increase the spread among the wider community, we developed a website which embeds the executable software and makes it easy to download.

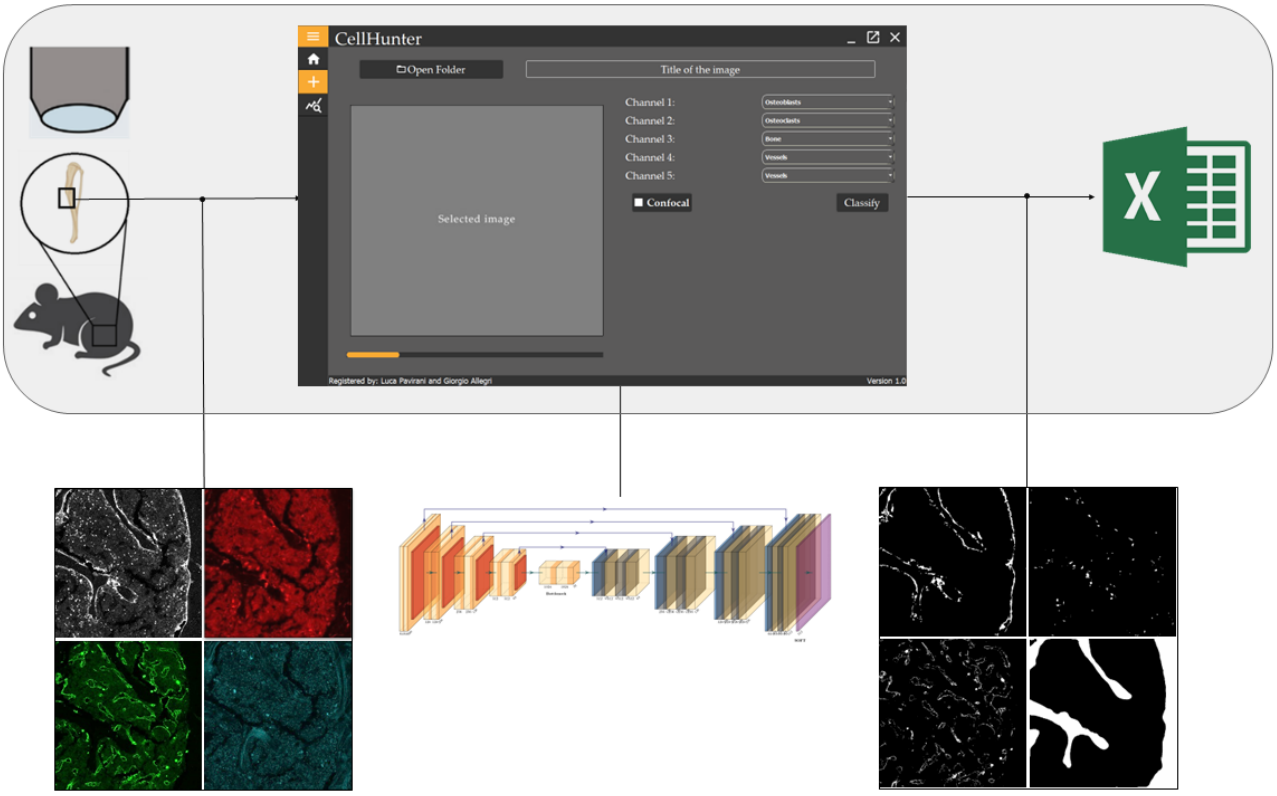


Figure 1: Pipeline of the work. From left to right: image acquisition process; automatic image segmentation; data quantification and extraction.

2. Materials and Methods

In this chapter, we present the complete process workflow, which consists of: i) dataset generation, ii) pre-processing, iii) dataset preparation , iv) neural network training, and v) post-processing with extraction of pivotal parameters, as illustrated in Figure 2. We conclude with a discussion of the software that embeds the neural network (NN) models.

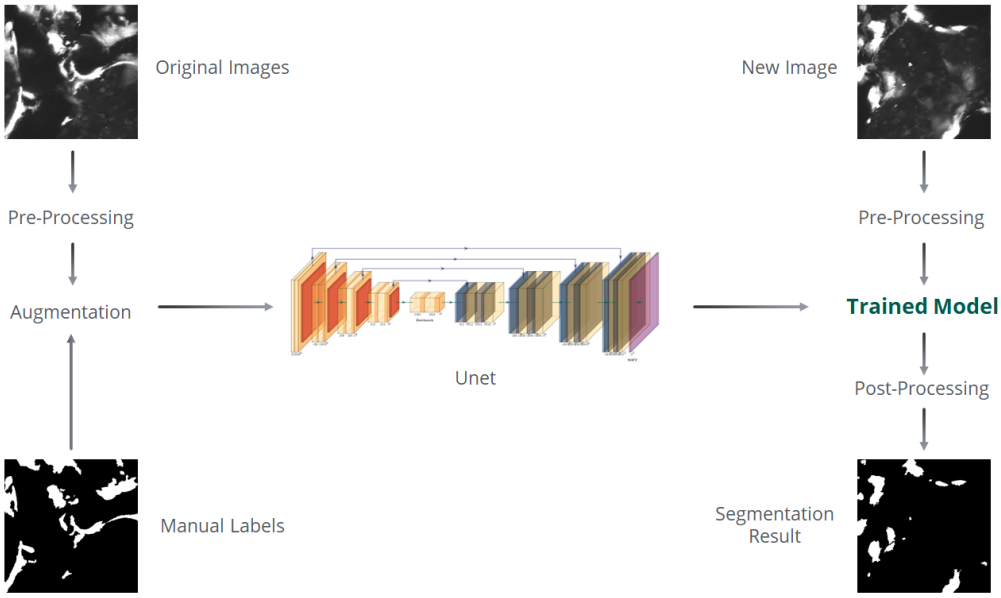


Figure 2: Workflow of the process.

2.1. Dataset Generation

The experimental dataset consists of bone mice images obtained from samples of mouse tibiae, calvaria, and spines collected by the GU Medical Oncology Department at the UT MD Anderson Cancer Center. The bones were first cleaned of tissues and muscles, and then sliced using a vibratome. The slices underwent a staining process, for which different antibodies were employed to highlight specific cell types, as shown in Figure 3. Finally, the slices were analyzed using either a custom Multiphoton Microscope (MPM) or a Confocal Microscope (CM).

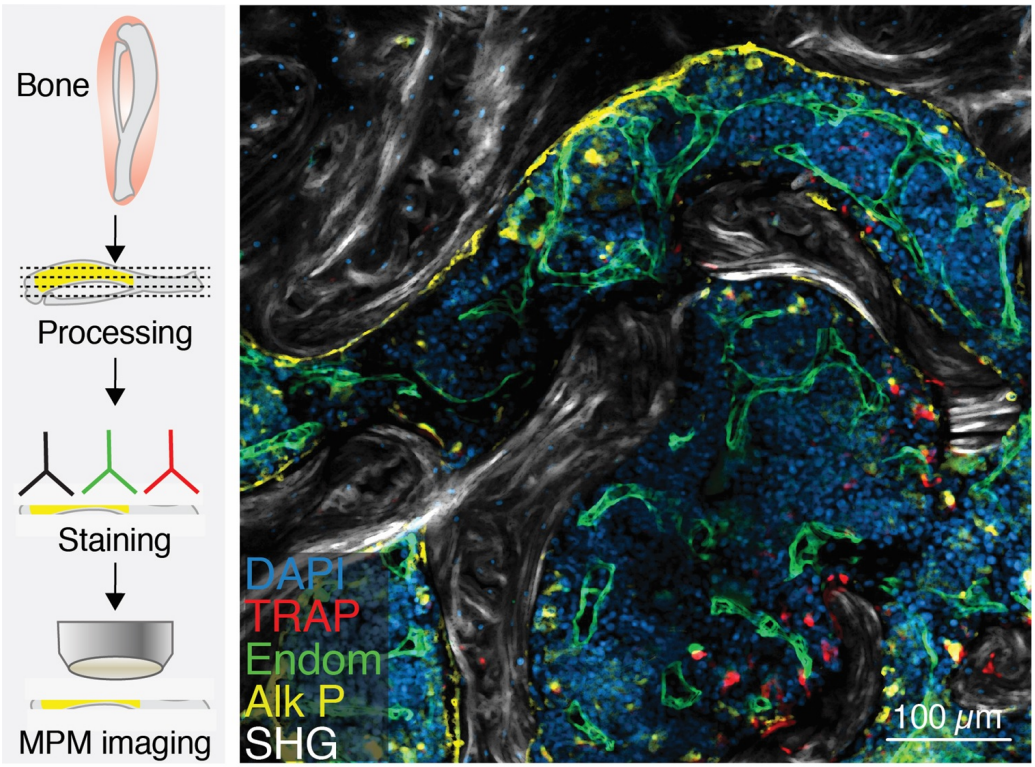


Figure 3: Sample preparation and image acquisition process.

The complete dataset is composed of a total of 175 volumes: 90 MPM and 85 Confocal stacks of images. Each volume consists of several channels, where each channel represents the fluorescence of an antibody linked to a specific cell type, as shown in Figure 4. Alkaline Phosphatase (Alk P) was used for OBs, tartate-resistant acid phosphatase (TRAP) for OCs, and endomucin for vessels. Concerning the bone structure channel, different methods were used for MPM and CM Images. In MPM acquisitions, bone information was identified in the Second Harmonic Generation (SHG) channel. In CM images, the representation of the bone channel is characterized by an absence of signal, making it unsuitable for segmentation purposes. To obtain bone information, the OBS and nuclei (4',6-diamidino-2-phenylindole - DAPI) channels were superimposed. Specifically, regions in which neither nuclei nor OBs were present were considered bone structure. This approach allowed for the characterization of the bone microenvironment in CM images despite the absence of a specific bone channel. To better highlight the structures of interest and their relative location respect to other cell types, we used the maximum projection of the volumes over the z-axis. Therefore, instead of obtaining an image from every single frame within the volume, a representative image characterizing the entire stack was obtained, as the information from the slices of the same stack might be redundant. Dimensions (depth, width, and height) of the acquired volumes are variable throughout the dataset. For the vessels' dataset, we used 148 slices from 7 CM volumes. We did not employ maximum projection for this structure because the information of the different slices is important to assess the correct shape and orientation of the vessels.

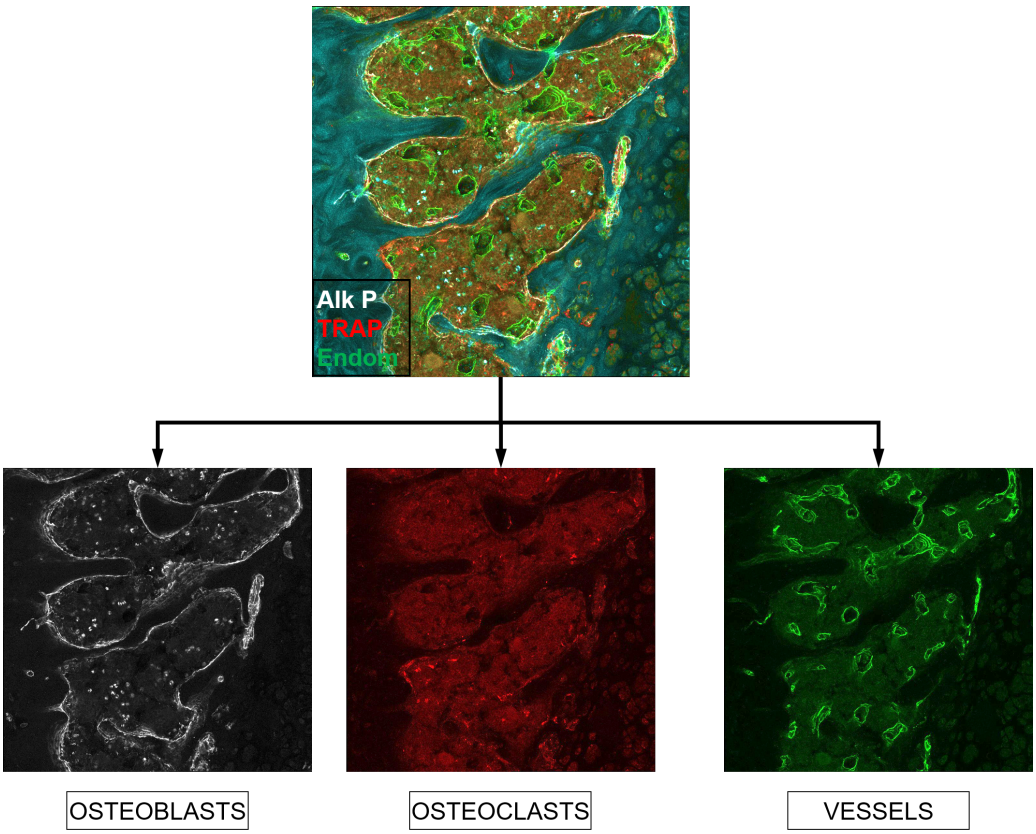


Figure 4: Example of a multi-channel confocal microscope image. OBs, OCs and vessels channels are shown.

The target masks (i.e. samples of model desired output) were created through manual segmentation using ImageJ software [23, 24]. Figure 5 shows an example of original image (left) along with its corresponding manually segmented mask. To process each image, we followed a set of steps customized for each dataset: the parameter values were empirically chosen after testing and analyzing their effect on the datasets' images. First, we adjusted the brightness and contrast to enhance features of interest visibility followed by the application of a Gaussian filter whose ratio was set empirically. Then, we used automatic (Otsu method [25]) or manual image thresholding to convert the image into a binary mask. However, the resulting mask was often affected by noise, so a denoising process was applied to improve the quality. Finally, the denoised mask were saved as ".tiff" image. Due to OC complex and variable structures, we optimized their mask accuracy by cropping the original images into smaller patches before following the aforementioned steps.

The total dataset counts 95 samples of OB channels, 75 OC and 90 MPM bone and 82 CM bone and 7 vessel volumes that resulted in 148 slices.

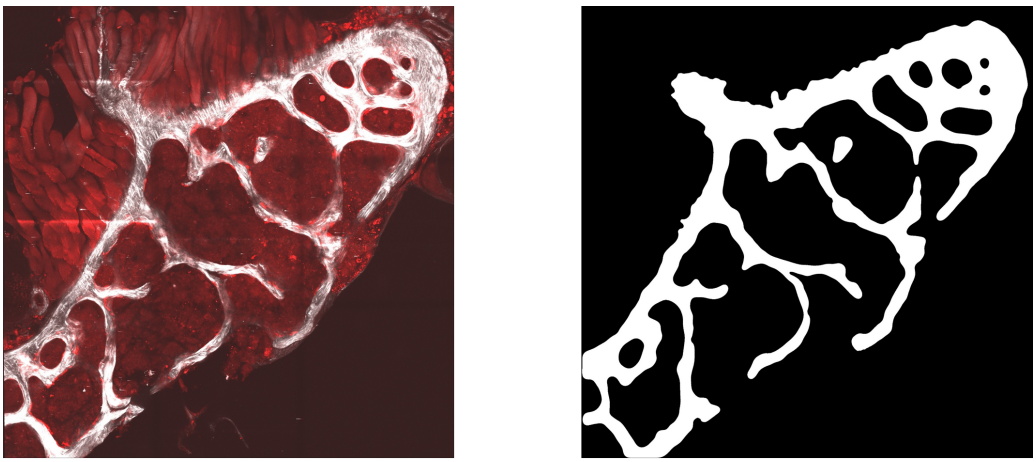


Figure 5: Original Bone Channel Image associated to its label.

2.2. Data Preprocessing

The images and corresponding labels underwent a series of preprocessing steps to optimize the DL algorithm training phase.

Pixel Normalization. Pixel values of all images were rescaled within the range of [0,1] to reduce fluctuations in the data (pixels' values) and to prevent poor algorithm performance.

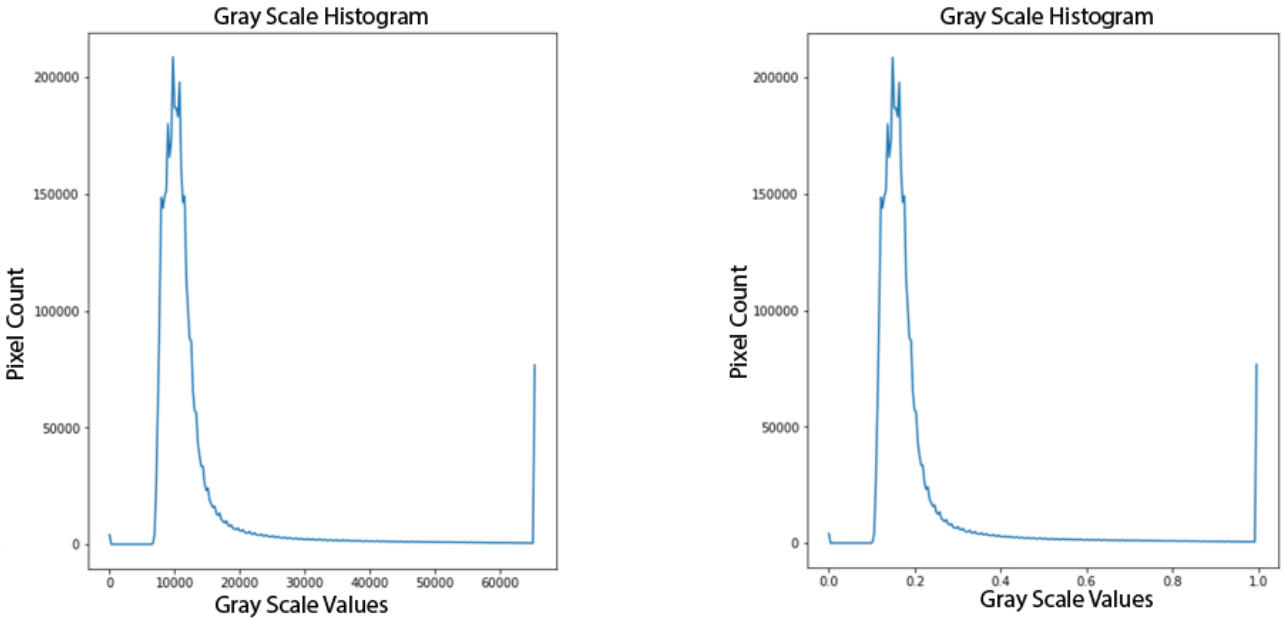


Figure 6: Normalization process. Plots show the pixel count (number of pixels with a certain value) against the gray scale pixel value before and after pixel normalization.

Gamma Transformation. This technique adjusts the brightness of the image by modifying the relationship between the input (original) pixel values and the output (transformed) pixel values by applying a power-law transformation as shown in Equation (1). For gamma values greater than 1, the image histogram shifts toward the left, and the output image results to be darker; on the other hand, for gamma values lower than 1, the image histogram shifts towards the right, and the output image results to be brighter. Table 1 shows the different gamma, corresponding to each structure, chosen to minimize the artifacts and to improve the structure design. This pre-processing step improves visibility of important features in the images and aids the algorithm to learn and differentiate between different structures.

$$V_{out} = V_{in}^{\gamma} \tag{1}$$

Histogram Equalization (stretching). A custom algorithm was implemented and used to distribute the intensity values over all the available range. The high and low levels with counts greater than a predefined threshold were remapped to 1 and 0, respectively. All other pixels were linearly remapped. For each cell type a different threshold was empirically chosen, and the resulting values are shown in Table 1. Along with our custom algorithm, used on the original dataset images, we applied an automatic histogram equalization function, known as CLAHE [26], to the augmented dataset images, to increase the variability of pixels' values, as we will explain in section 2.3.

Black Image Removal Images with low number of white pixels were removed to balance the dataset and improve the training speed of the algorithm. These images were removed along with their labels since they did not provide any significant information for the algorithm training. Consequently, we reduced the number of black pixels, obtaining a more balanced dataset since an imbalanced dataset can lead to inaccurate predictions and suboptimal results. Therefore, this step was necessary to ensure the efficiency and accuracy of the algorithm.

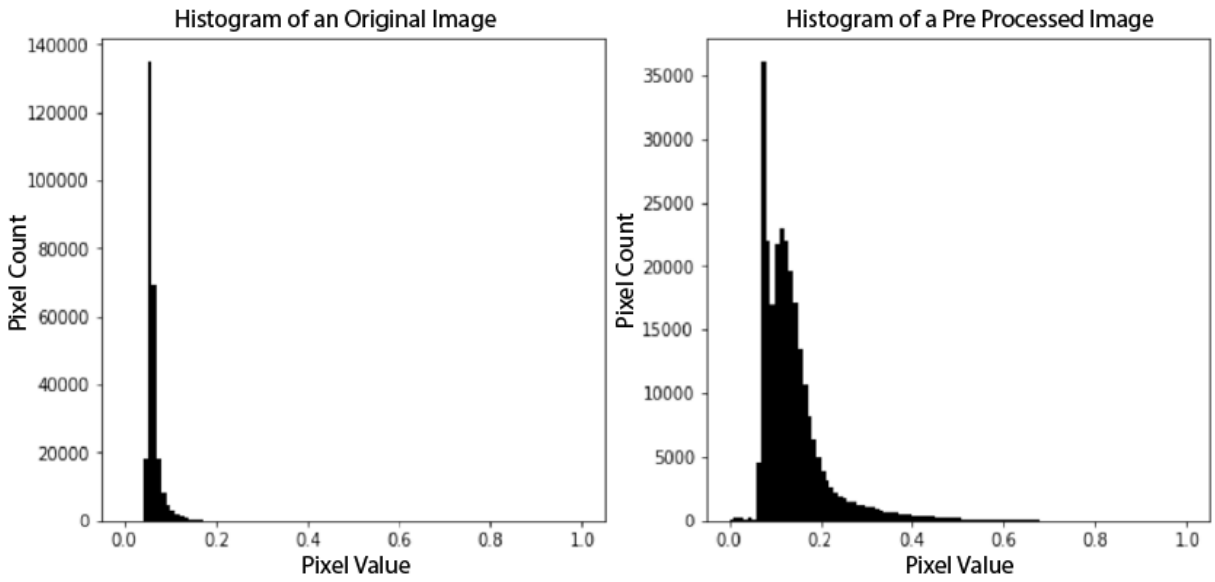


Figure 7: Histogram Stretching process. Pixel count plotted against pixel intensity value before (left) and after (right) histogram stretching. The range of assumed values increases and the quality of the image improves.

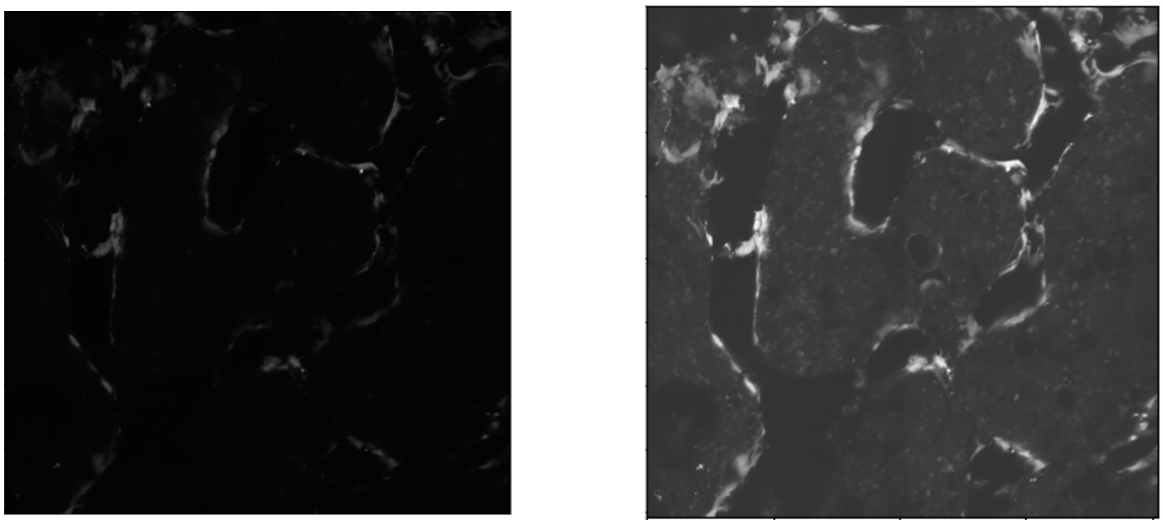


Figure 8: Original image (left) against preprocessed image (right) after the application of gamma transformation and histogram equaliztion.

Channel	Gamma	Threshold
OBs	1	100
OCs	0.75	150
Bone	2	250
Vessels	1.5	200

Table 1: Empirically chosen gamma and threshold values.

Sample-wise pixel weighting. Each pixel was assigned a weight to achieve an optimal separation of close or adjacent cells, as suggested by Ronneberger et al. [27]. Each mask was associated to a map that assigned a specific value to each pixel. This procedure was applied to the OC dataset since those cells were the most complex to be correctly detected. Pixels belonging to OCs were set to 0, while the background pixels were weighted depending on their distance from the two nearest cells. Therefore, the lower the distance, the higher the weight, and more importance was given to the correct classification of that pixel in the training phase. The pixel weighting result is shown in Figure 8, where blue is assigned to less important (i.e. higher distance) pixels, while red represents the most important ones (i.e. lowest distance). Each pixel $x_{i,j}$ (in position i, j) was

assigned a particular weight ($w_{i,j}$). To restrict the weights' magnitude, their value was computed by applying a Gaussian function to the Euclidean distance ($D_{i,j}$), as it follows:

$$w_{i,j} = \begin{cases} 0 & \text{if } x_{i,j} \notin \text{background} \\ ae^{-\frac{D_{i,j}}{2\sigma^2}} & \text{otherwise} \end{cases} \tag{2}$$

$$\text{where: } D_{i,j} = (d_1(x_{i,j}) + d_2(x_{i,j}))^2 \tag{3}$$

In Equation 2, $w_{i,j}$ represents the weight of the pixel at position (i, j), a is the amplitude of the Gaussian, empirically set to 100 and σ, the standard deviation, was set to 5 pixels. In Equation 3, $d_1(x_{i,j})$ and $d_2(x_{i,j})$ are the distances of the pixel in position (i, j) from the 2 closest cells.

Figure 9: From left to right: original image, segmented mask, pixel-wise weight map. Blue means less importance; red represents more important pixels.

2.3. Data Preparation

The input images were labeled with masks that serve as a reference during model training. To detect a single type of object, two classes were used: one to represent the object of interest, and the other to represent the background. In this application, pixels that corresponded to OCs, OBs, bone, and blood vessels were assigned a value of 1 (white), while pixels corresponding to the background were assigned a value of 0 (black). Using this labeling approach, the NN is able to learn to differentiate between the object of interest and the background, leading to segmentation of the input image. All the steps explained here were performed for every cell structure dataset (OBs, OCs, Bone CM, Bone MPM and Vessels).

Train-Validation split. A training and a validation set were generated splitting the dataset in 80% training and 20% validation. This process was accomplished using the python scikit-learn library [28]. The training subset was used to train the model, while the validation subset was used to provide an impartial evaluation of the model performance on the training set, allowing for network parameter tuning. To ensure no repetition of information between the training and validation datasets, the augmentation was applied after data splitting. Moreover, to increase the variability in the dataset and prevent overfitting, a mixed preprocessing algorithm was also applied. This involved randomly combining the results of different preprocessing techniques, such as histogram stretching and gamma transformation, on each augmented image. This allowed the algorithm to learn from a wider range of different image characteristics.

Image2patch. Due to computational constraints, the original size of the images could not be used to train the algorithm. To overcome this limitation, the images were cropped to a fixed size instead of being resized via interpolation, which can change important pixel values for the DL algorithm. We found significant pixel loss while using patching libraries available on PyPI (e.g. "Patchify" [29]). Accordingly, we developed a custom patching algorithm called "image2patch" [30], now available on PyPI. This algorithm minimizes pixel loss during the patching process by adapting the patch stride to the original dimensions of the image creating small overlap, hence working also as an augmentation process. With this approach, tens of patches can be retrieved from one original image, providing a large amount of data for the deep learning algorithm to learn and make predictions. The patch size was set to 512x512 pixels for all the performed trainings. An example is shown in Figures 10 and 11.

Data Augmentation. To further improve model performance, data augmentation techniques were employed to increase the dataset size, feeding more images in the NN. The applied transformations included discrete

rotations of $\pi/2$ and transposition to the original images. Moreover, a random pre-processing step (automatic histogram equalization or custom pre-processing with different values of gamma and threshold) was applied depending on a probability factor. This process aimed at generating new data and increasing the variability within the dataset. In this way, the NN was trained on a wide range of pixels values becoming insensitive to different brightness and contrast levels that are highly dependent on the acquisition process. No elastic deformations were applied to the images to avoid the invalidation of the cells' shapes, to which the algorithm should not be invariant as these are relevant factors. The Albumentations library [31] was tried to employ additional random techniques for data augmentation, but it did not result in any significant improvements. Therefore, the decision was made to stick with the simpler rigid geometrical transformations and contrast/brightness variations as previously described. A visual representation of these transformation is shown in Figure 12. Table 2 shows the final number of images for the training and the validation datasets after all the preparation steps.

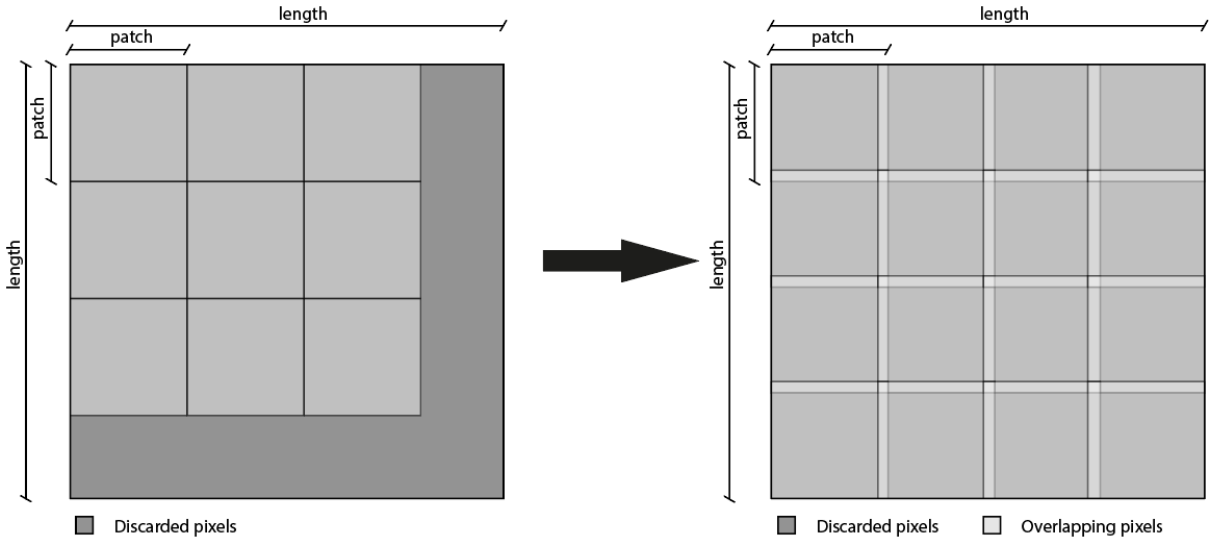
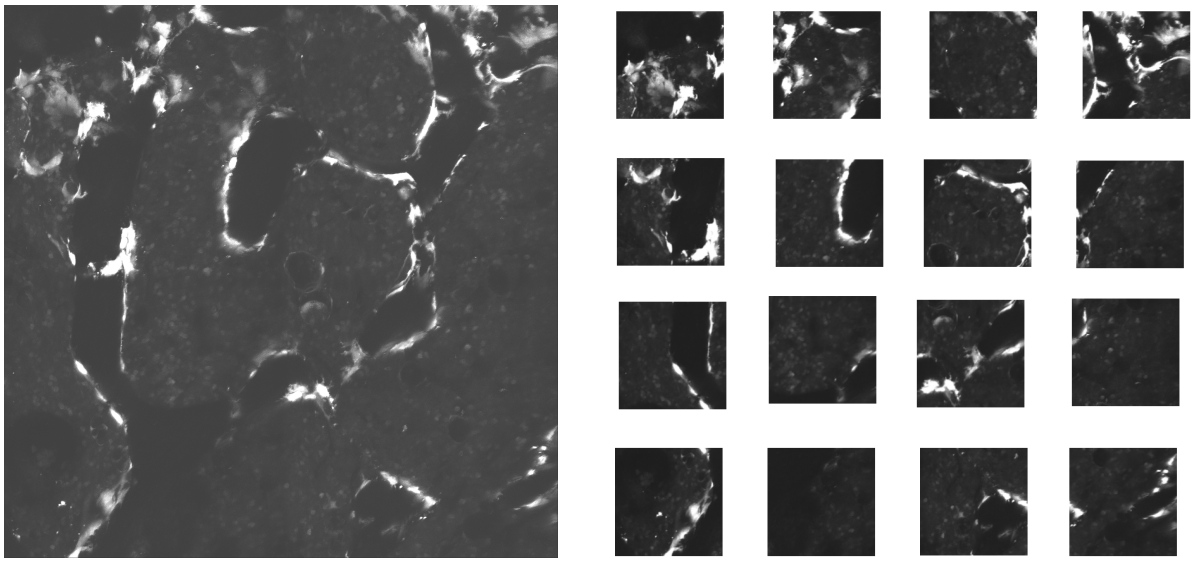


Figure 10: On the left is shown the previous patching algorithm that did not provide adaptive patching step calculation, leading to a large amount of lost pixels (in dark gray). On the right is shown the image2patch algorithm. It adapts the patch step in order to minimize pixels' loss.



(a) (b)

Figure 11: 11a: Original Image. 11b: from the original image, 16 smaller patches are retrieved.

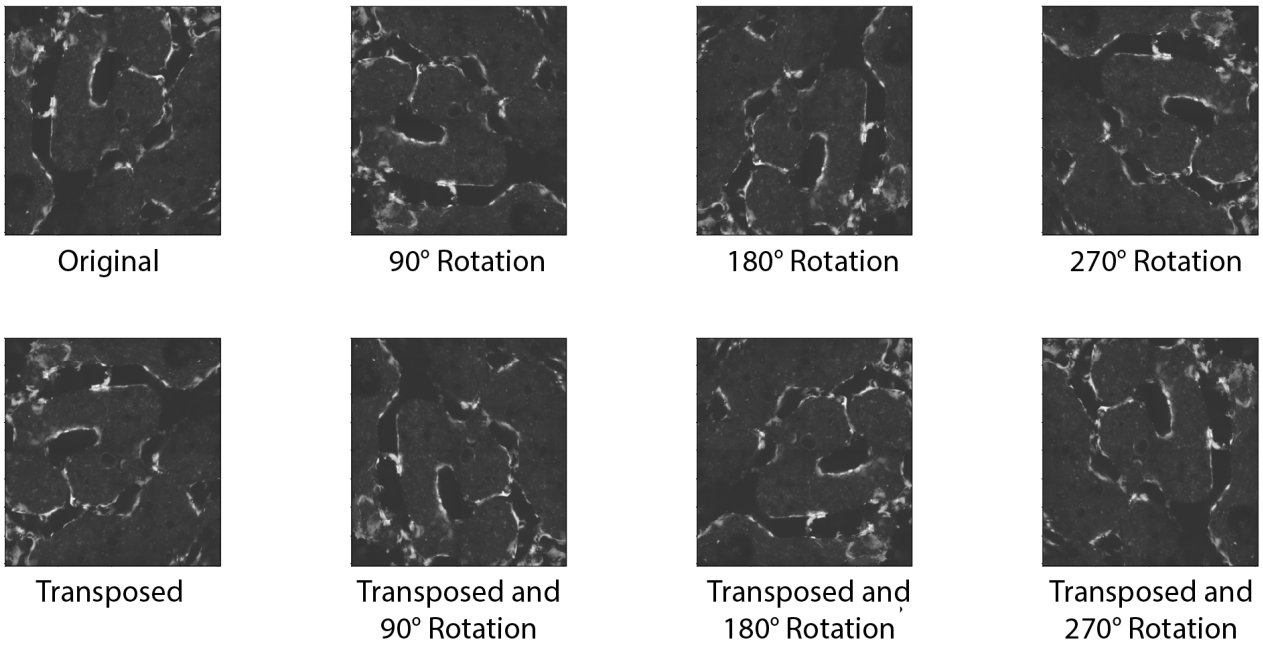


Figure 12: Augmentation process.

Cell Type	Training	Validation
OC	3840	768
OB	3542	849
Bone MPM	3728	786
Bone CM	2904	1248
Vessels	3090	640

Table 2: Number of images after the pre-processing steps (black image removal, image patching and data augmentation) for the training and the validation datasets.

2.4. Model Training

Neural Networks. The dataset was prepared to optimize the NN training. DL is a branch of Machine Learning (ML) that uses NNs to process unstructured data through a self-learning strategy. A NN consists of input, hidden, and output layers that work together to make predictions. When data is fed into the input layer, it is processed through the hidden layers using weighted connections between the neurons. These weights are adjusted during training to improve the accuracy of the predictions made by the network. Through this process, the NN can independently learn to recognize and extract patterns from the data. The strength of DL lies in its ability to handle unstructured data, such as images, audio, or text, by allowing the network to automatically extract features and patterns from the input data. This makes it a powerful tool for a wide range of applications, from image and speech recognition to natural language processing and predictive modeling.

Fully Convolutional Networks (FCNs). FCNs are a type of DL model commonly used in computer vision applications. One of the defining features of a FCN is the inclusion of only convolutional layers, which utilize filters that move across the input data. These filters, also known as kernels, traverse the input data in a horizontal and vertical manner, column by column and row by row, respectively. Through this process, FCNs can extract meaningful features and patterns from the input data, making them a powerful tool in the field of computer vision. A general example of a FCN structure is shown in Figure 13. The input data is fed into the network from the left side, then undergo filtering and processing and the predicted label is computed in the last layer on the right.

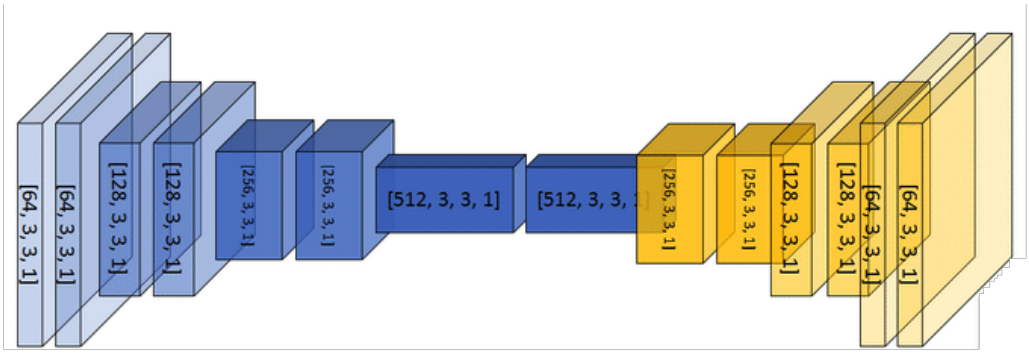


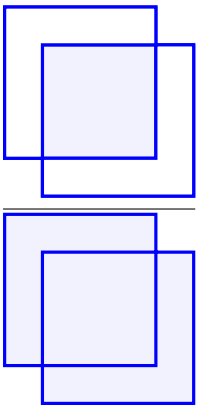
Figure 13: Fully Convolutional Network scheme. Data flows from left to right through convlutional layers.

Our FCN models were first built and trained using Google Colaboratory, which presented certain computational limitations, such as a maximum of 25GB of RAM, a runtime of up to 12 hours, and either a T4 or P100 GPU. The most restrictive limitation was related to the RAM, which necessitated a significant reduction in the image dimensions in order to accommodate the available memory. Additionally, to fit the image sequences into the FCN model, each image needed to share the same dimensions. To further meet memory limitations, we organized our dataset in small batches and fed these directly to the model. By addressing these limitations and optimizing the model to work within the constraints of the available resources, the FCN was able to be trained effectively and generate accurate results. The final models were trained using a local workstation with a GPU RTX A 5000 and a 128GB DDR4 RAM, which allowed faster trainings with a higher amount of data fed in once to the model.

A particular FCN structure is shown in Figure 14, the U-Net [27]. U-Net is a FCN that follows a U-shaped architecture and consists of two quasi-mirrored Convolutional NNs (CNNs). The first path, called the encoder, is a contraction path that uses convolutional and max-pooling layers to down-sample the input image. This process helps to increase the receptive fields and obtain information about the image content. The second path, known as the decoder, is an expansion path that uses convolutional and transposed convolutional layers to up-sample the image and obtain information about the position of objects. The two paths interact through "skip-connections", represented by the blue arrows in Figure 14, that allow the concatenation of the output of each encoding layer (contraction path) to the same decoding layer in the upsampling path. After each concatenation, two consecutive convolutions are performed to combine the information obtained from both paths. The last layer consists in a sigmoid activation function that ultimately gives, to every pixel, the probability to be either 1 (pixel belongs to a cell) or 0 (pixel belongs to the background).

To build, train and validate the model, we used the following Python libraries: TensorFlow [32], Keras [33] and Segmentation Models [34]. The evaluation of the models was done considering classic statistical metrics, as Accuracy, Precision Recall and the Intersection over Union (IoU) metric. IoU is commonly used to measure the performance in object detection tasks and it is calculated by dividing the intersection between the ground truth and the predicted mask by the union of the two.

$$IoU = \frac{\text{area of overlap}}{\text{area of union}} = \quad (4)$$



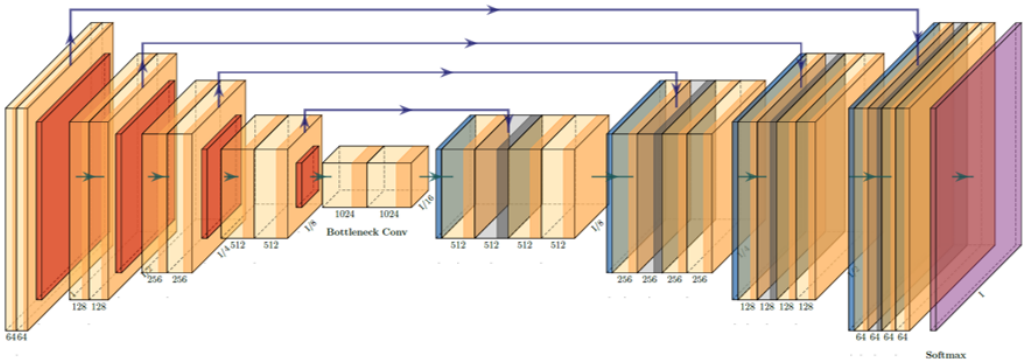


Figure 14: U-Net architecture. Encoder path: the image goes through a set of convolutional and max-pooling layers (green arrows). Decoder path: the image is reconstructed through up-convolution layers (green arrows). Skip connections (blue arrows) allow for the integration of the information from the down-sampling layer to the same layer of the up-sampling path.

Hyperparameters Tuning. Hyperparameters tuning is an essential step in the development of DL models, as it allows optimizing the model performance on the specific task at hand. In our bone stromal cell segmentation task, we focused on tuning several U-Net hyperparameters:

- *Learning rate:* controls the step size at each iteration in the optimization process during training. A high learning rate can favor fast convergence, but it can also cause the global minimum overshooting, hence, never converging. Conversely, a low learning rate can result in stable training, but it can also lead to slow convergence, and it could get the model stuck in a local minimum. A sigmoid-like learning rate function was implemented as shown in Equation 5 and the resulting behaviour is plotted in Figure 15. In this way, both fast convergence and stable training are reached.

$$lr(E) = \begin{cases} (lr_0 - lr_\infty) \times \frac{1}{1+e^{s(E)}} + lr_\infty & \text{if } E < E_{threshold} \\ lr_{plateau} & \text{if } E \geqslant E_{threshold} \end{cases} \tag{5}$$

$$\text{where: } s(E) = \frac{E - \frac{E_{threshold}}{2}}{\frac{E_{threshold}}{8}}$$

- *Batch size:* determines the number of training samples used in each iteration of the optimization process. A large batch size can result in fast convergence and in smooth optimization. However, it requires more memory, which is not always available, and it could get stuck in a local minima. Conversely, a smaller batch size leads to a more stable optimization and a better generalization, but is characterized by slower training.
- *Number of epochs:* determines the number of times the entire training dataset is passed through the network during training. A high number of epochs can result in better performance, but can also lead to overfitting and longer training time. Conversely, a low number of epochs can result in faster training but can also lead to underfitting.
- *Lambda regularization:* is used to prevent overfitting by adding a penalty term to the loss function that keeps the network weights small. A higher lambda value can result in a strong regularization preventing overfitting, but can also lead to underfitting.
- *Number of filters*: determine the complexity of the U-Net architecture, affecting its performance. A high number of filters can result in a deeper and more powerful network but increases the risk of overfitting and requires more memory. Conversely, a low number of filters can result in a simpler network but may not be powerful enough perform complex tasks.

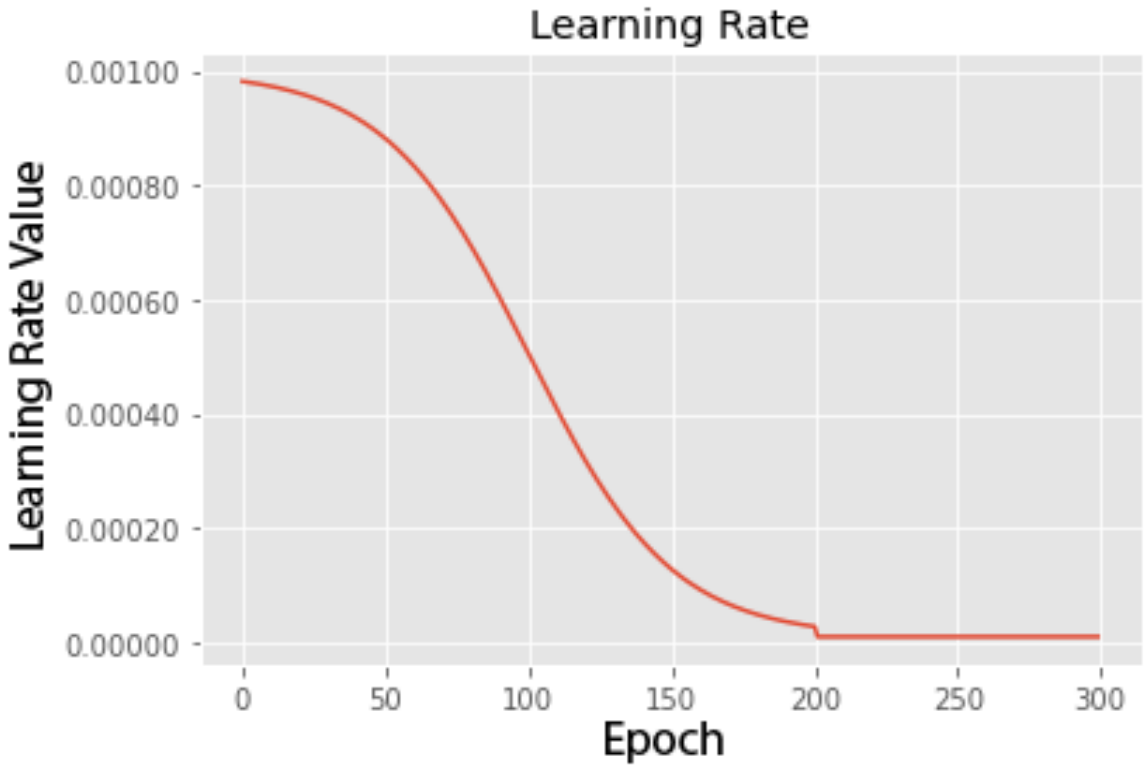


Figure 15: Learning rate values are plotted against the epoch. At the threshold epochs the learning rate is set to the plateau value.

To find the best hyperparameters for our specific task (bone stromal cell segmentation), we used the Keras Tuner Search tool [35], performing a randomized search over a predefined hyperparameter space. This allowed us to efficiently explore a wide range of hyperparameter values and find the optimal combination returning the highest segmentation accuracy. Consequently, for each cell type, different hyperparameter values were chosen as shown in Table 3.

Cell Type	Learning Rate	Batch Size	Epochs	λ Regularization	Number of Filters
OBs	0.0005	32	300	0.01	28
OCs	0.001	16	300	0.0005	32
Bone MPM	0.00005	32	300	0.005	28
Vessels	0.001	16	300	0.005	16
Bone CM	0.0001	24	300	0.001	32

Table 3: Hyperparameter values. The "learning rate" refers to the central value of the sigmoid function. The Number of Filters is the number of convolution filters in the top layer.

Loss Function. In the study, the loss function used to guide the deep learning models in their training is the Tversky Loss [36]. This loss function penalizes false positives (FPs) and false negatives (FNs) by assigning weights to the error through two parameters, alpha (α) and beta (β). These values were determined experimentally. Additionally, to address the issue of class imbalance, the loss function is further weighted based on class-specific weights. The loss function is minimized by applying Adam Optimizer [37]. Other common loss functions, such as Binary Cross Entropy Loss and Dice Loss were tested, but they were not able to address the problem as effectively as the Tversky Loss [38].

The architecture first employed in this application is a modified version of the original U-Net described in [27]. The customized U-Net was originally tailored for semantic segmentation in the field of foreign body response [20]. This custom structure was selected due to its effectiveness, low computational needs, and because the input images captured with the same multiphoton microscope had similarities to those in the current study. Hyperparameter tuning was applied to this version of the U-Net for every cell type (Table 3). To further improve the segmentation results, particularly for OCs, that are characterized by the most variable structures

and pixel values, we tested different architectures: a U-Net++ [39], a ResNet [40] and a U-Net with ResNet34 as backbone [41]. The segmentation results in terms of IoU over the OC dataset are shown in table 4, and, clearly, the last implemented and tested network is the best performing one, hence we used it also for the training and segmentation of the other cell-types.

Architecture	Validation IoU
Custom U-Net	0.48
U-Net++	0.46
ResNet	0.39
U-Net - ResNet34	**0.57**

Table 4: IoU validation values for different architectures over the OC dataset.

In our study, the loss function used to guide the DL models in their training is the Tversky Loss (TL) [36]. This loss function penalizes false positives (FPs) and false negatives (FNs) by assigning weights to these errors through two parameters, alpha (α) and beta (β). Additionally, to address the issue of class imbalance, the loss function was further weighted based on class-specific weights. The loss function was minimized by applying Adam Optimizer [37].

$$TL_c = \sum_c (1 - TI_c) \tag{6}$$

$$\text{with} \quad TI_c = \frac{\sum_{i=1}^{N} p_{ic} g_{ic} + \epsilon}{\sum_{i=1}^{N} p_{ic} g_{ic} + \alpha \sum_{i=1}^{N} p_{i\hat{c}} g_{ic} + \beta \sum_{i=1}^{N} p_{ic} g_{i\hat{c}} + \epsilon}$$

where the TL for class c (a certain cell type) is calculated based on the Tversky index (TI). p_{ic} is the probability of a the i-th pixel to be of class c, while g_{ic} represents the ground truth value for that pixel. In the same way $p_{i\hat{c}}$ and $g_{i\hat{c}}$ refer to the probability and ground truth fot the background class. N is the total number of pixels in the image. The parameters α and β control the relative weighting of false positives and false negatives in the loss function. This function penalizes false positives and false negatives differently depending on the values of α and β. For example, if $\alpha > \beta$, the loss function will place more emphasis on reducing false positives, while if $\beta > \alpha$, it will place more emphasis on reducing false negatives. α and β were chosen empirically and are shown in Table 5.

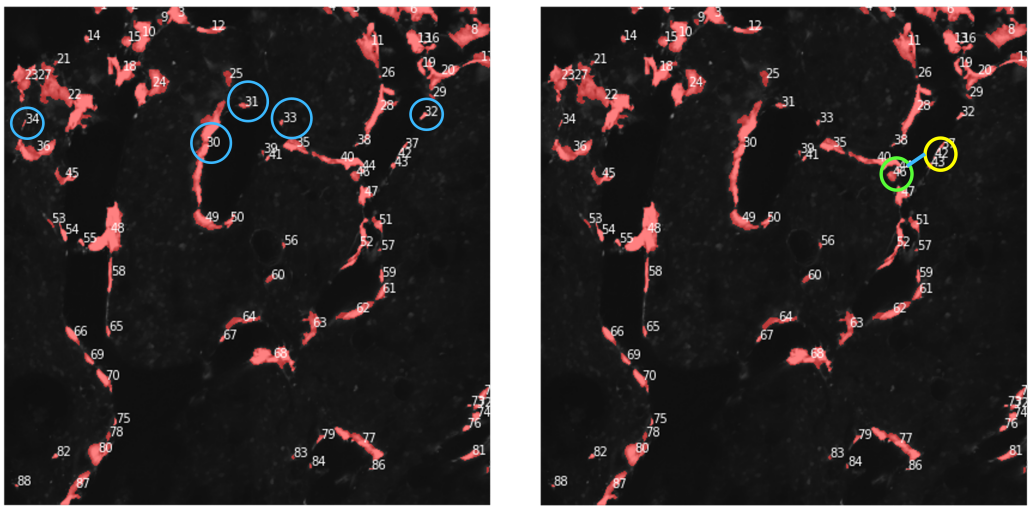
2.5. Post Processing

In the post-processing phase, parameter extraction was performed in Python environment using both custom functions and built-in libraries to retrieve measurements of interest on each cell type.
Regarding OC, we extracted cell count, location, inter-cell distances, distance from the closest OB, and their coverage of the bone contour. To determine the number of OCs, 8-connected structures were classified as cells if their area exceeded a predetermined threshold. The location of each OC was determined by calculating the region center of mass with coordinates defined along the X and Y directions. Inter-cell distances were calculated based on the relative positions of the OCs as the Euclidean distance between the coordinates of the centroids. Distances exceeding a certain threshold were deemed irrelevant and were not considered. The OC-OB minimum

Training Model	α	β
OB	0.3	0.7
OC	0.3	0.7
Bone MPM	0.3	0.7
Bone CM	0.3	0.7
Vessel	0.7	0.3

Table 5: α and β chosen for the Tversky Loss in every trained model. The cell-types represent the dataset of cells for which the models were trained.

distance was calculated as the Euclidean distance between the OC centroid and the closest point on the line of OB.



(a) OCs Count. (b) Inter-OC distances.

Figure 16: Cell count and inter-cell distances.

Due to the line-like appearance of OBs along bone structure contour, it was not possible to extract the same parameters as for the OCs. Instead, bone coverage was calculated as the percentage of the common area between the OBs and bone contour, relative to the bone contour alone, as shown in equation 7.

$$\text{Cell Bone Coverage} = \frac{\text{Common Area}}{\text{Bone Contour}} \quad (\%) \tag{7}$$

The bone contour and the OB prediction were not always perfectly overlapped. This led to a coverage underestimation, since most of the OB region was not counted as covering the bone contour. For this reason, a morphological transformation, named dilation, was applied to the bone contour with a 3x3 structuring element. Equation 8 describes the math behind the process. This increased the contour width and improved the bone coverage calculation accuracy. The dilation of A by B is defined by:

$$A \bigoplus B = \bigcup_{b \in B} A_b \tag{8}$$

where A_b is the translation of A by b.
OC predictions might be affected by false positive in regions with bright cells that are not OCs. To overcome this limitation, OC bone coverage was calculate following the same algorithm and calculations.

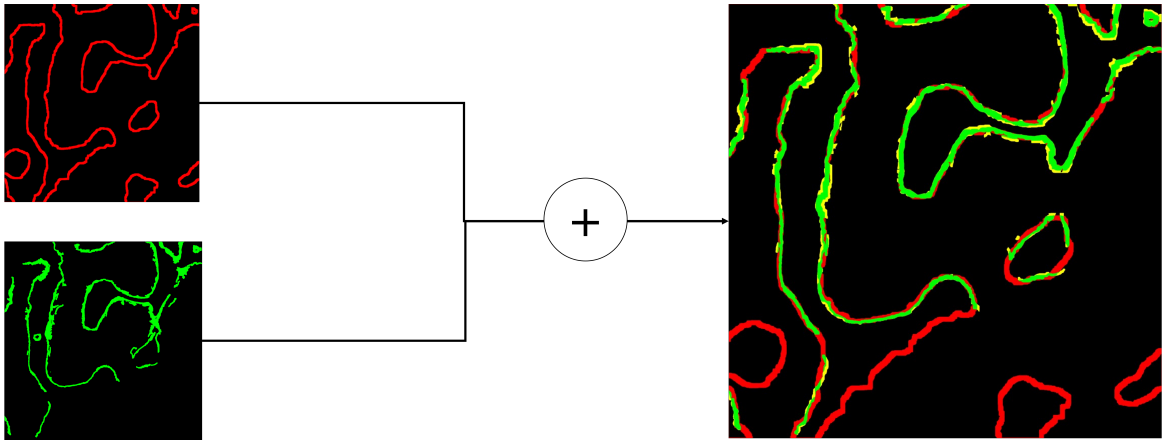
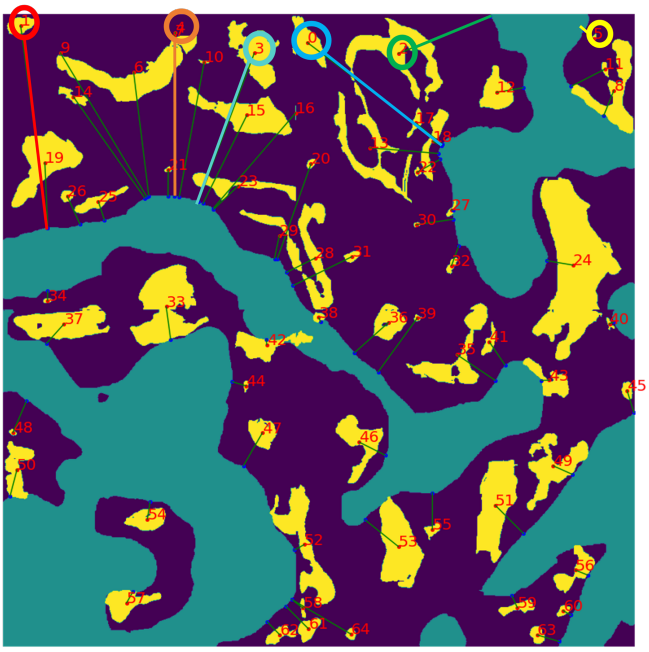


Figure 17: OB Bone Coverage. In red the dilated bone contour, in green the predicted osteoblast mask. The algorithm superimposes the two channels and calculates the cell bone contour.



Vessel #	Distance (px)
0	256.209
1	310.022
2	148.981
3	240.811
4	249.002
5	18.536

Figure 18: Vessel - Bone distance. The yellow portions represent the segmented vessels, while the blue ones correspond to the bone structures. The green lines represent the segment between a vessel centroid and the closest point on the bone surface. The right table shows the first 6 measurements, highlited in the left image.

Additionally, the number and location of vessels were extracted, along with the euclidean distance between each vessel and the nearest part of the bone, as shown in Figure 18. The extracted parameters are essential for quantifying the effects of therapies on the bone microenvironment, the response to bone metastases and their impact on bone cells. For example, in osteolytic bone metastasis, the number of OCs increases, particularly in regions close to the metastasis. The increase can be accurately quantified by counting the number of cells within a specific region of interest. Conversely, the number of OBs decreases, resulting in decreased bone coverage. This variation can be assessed by calculating the portion of the bone contour covered by OBs.

2.6. Software

The purpose of the GUI is to provide an intuitive and user-friendly interface for biologists to interact with the models and easily perform segmentation tasks. The development of the GUI involved the use of Adobe Illustrator, Qt Designer, and PyQt5 Python library. In Figure 19 the final layout of the user interface is shown.

Adobe Illustrator. Adobe Illustrator [42] is a powerful vector graphics editor that is widely used for graphic design, digital illustration, and user interface design. In the context of our work, Adobe Illustrator was used to create a visually appealing design for the GUI. The design process in Adobe Illustrator involved creating sketches and wireframes of the GUI to establish the overall look and feel, as well as the placement of buttons, text fields, and other elements. The vector graphics capabilities of Adobe Illustrator allowed for the creation of high-quality graphics and illustrations that could be easily scaled and adjusted as needed. Once the design was finalized, it was exported as an image file and used as the foundation for the implementation of the GUI using Qt Designer.

QT Designer. Qt Designer [43] was utilized to create a variety of GUI elements such as buttons, text fields, and sliders, and to position them on the screen in accordance with the interface design. The software also provided tools for customizing the appearance of these elements, such as adjusting font styles, color schemes, and button shapes. To ensure the functionality of the interface, Python code snippets were integrated into the Qt Designer implementation process. These code snippets were used to define the behavior of specific GUI elements, such as the actions triggered when a button was clicked, or a slider was moved.

PyQT5. PyQT5 Python library [44] was used since it provides a range of functionality for the GUI, including the ability to create signals and to trigger commands in response to the user interacting with the GUI elements such as buttons and sliders. The PyQT5 library was particularly useful in ensuring the responsiveness of the interface, as the ability to trigger commands in real time allowed for a smooth and seamless user experi-

ence. The library also offered a variety of customization options, such as the ability to adjust the appearance of the GUI elements and to incorporate animations and transitions into the interface design.

The resulting GUI provided biologists with an intuitive and visually appealing interface to interact with deep learning models, which met all the functional requirements.

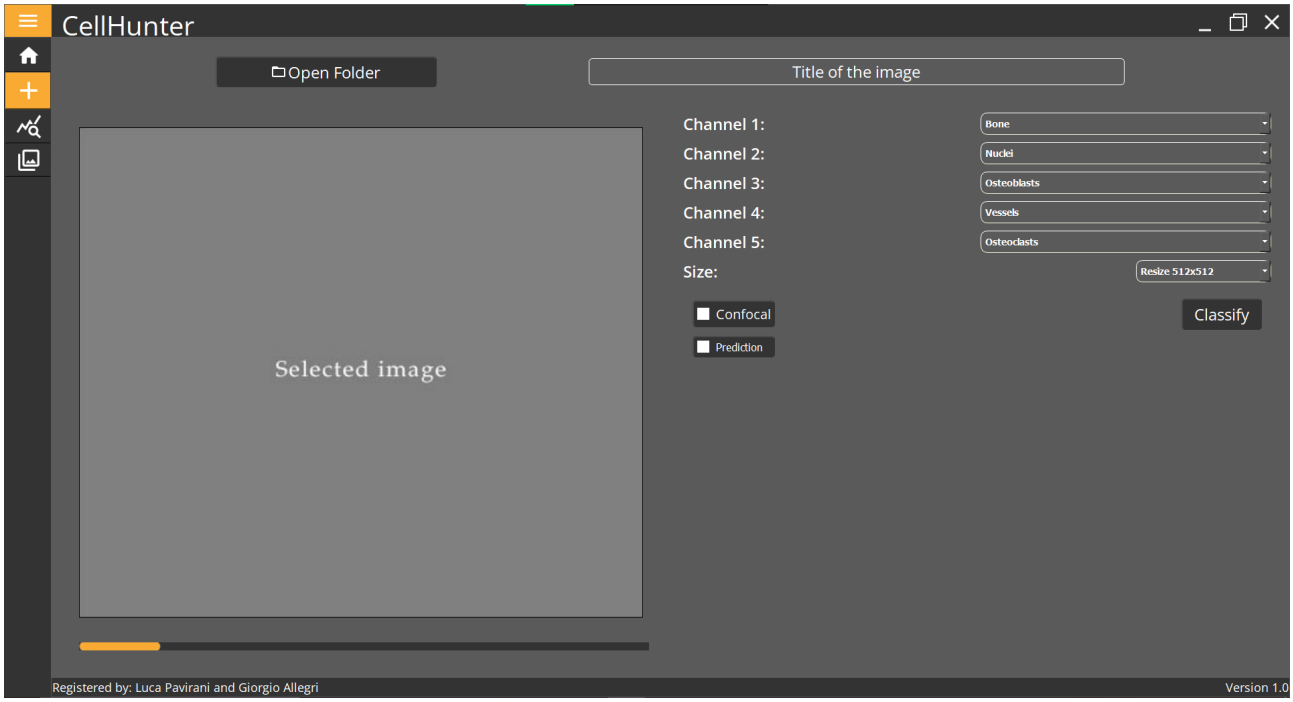


Figure 19: CellHunter GUI.

2.7. Website

To make our deep learning models for bone stromal cell image segmentation more accessible to biologists, we developed a GUI in Python that has been converted into an executable software. This software can be easily downloaded and used without any knowledge of programming languages.

To increase the visibility of the software and make it accessible to a wider community of researchers, we developed a website that provides comprehensive information on the software, its features, and its usage. The website includes a logo, as well as information on how to download and install the software. The website also includes a section that showcases some of the images used in the training of the models, providing users with an opportunity to see how the software works in action. Additionally, there is a section that explains how to use the software, which is helpful for biologists who are new to the field of deep learning and image segmentation. To further enhance the usability of the website, we included a review section where users can leave feedback on their experiences using the software. This feedback helps us to improve the software and make it more user-friendly.Furthermore, we have also provided links to our contact information, including our address, LinkedIn, and GitHub profiles, in case users have any questions or need further assistance with the software.

As a future development, we plan to add a feature that will allow users to fine-tune the models using their own images, which will increase the accuracy of the software and make it more versatile. This will allow researchers to tailor the software to their specific research needs and further expand its utility in the field of bone stromal cell analysis.

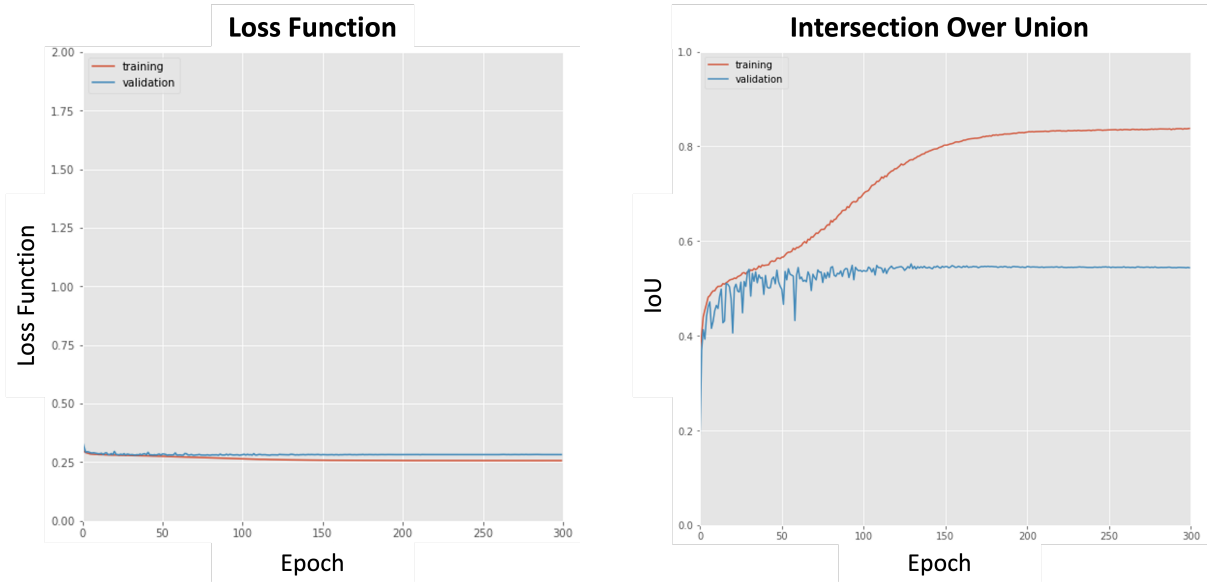
3. Results and Discussion

This chapter presents and discusses the results of the U-Net based semantic segmentation of bone stromal cells, with attention to the choices of model, segmentation results and comparison with ground-truth mask validated by expert biologists.

3.1. Segmentation Model Training

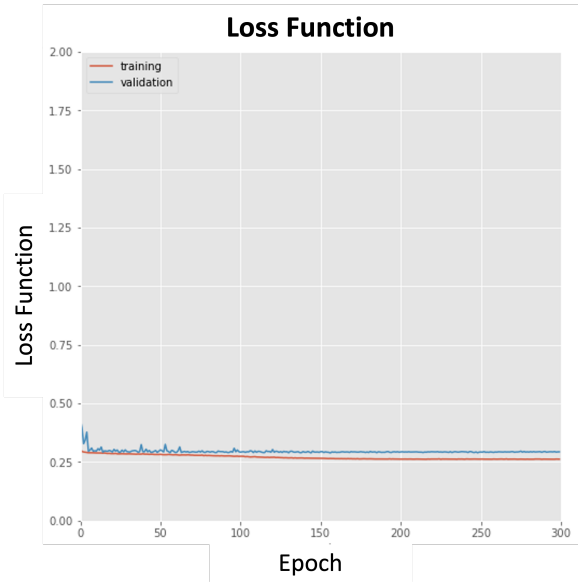
The model was trained and validated with images from different sources (MPM and CM) to increase the networks' ability to learn from different types of microscopic images. The images were manually labeled to discriminate the different structures in each image. After preprocessing pipeline, explained in Section 2.2, data were split into training and validation sets with 80% and 20% portion respectively and augmented by a factor of 8, thus maintaining the two datasets independent of each other to avoid possible repetitions of information. We tuned the model hyperparameters with Keras Tuner, as shown in Section 2.3, with results reported in detail in Table 3. We empirically chose the best combination of parameters α and β for the Tversky Loss, as shown in 5.

Semantic segmentation performances in the training phase were monitored and evaluated in terms of Intersection over Union (IoU), as mentioned in Section 2, which returns an absolute metric value as the ratio between intersection area and union area of predicted and target foreground mask. Ideally, the best reachable asymptotic value for this metric is 1. The other fundamental parameter is the loss function that during the training phase needs to be minimized. The training results for the different cell types, in terms of IoU and Loss Function are shown in Figures 20 (Osteoblasts), 21 (Osteoclasts), 22 (CM Bone structures), 23 (MPM Bone structures) and 24 (Vessels), where the training (red line) and validation (blue line) Loss Function and IoU are plotted against the epoch.

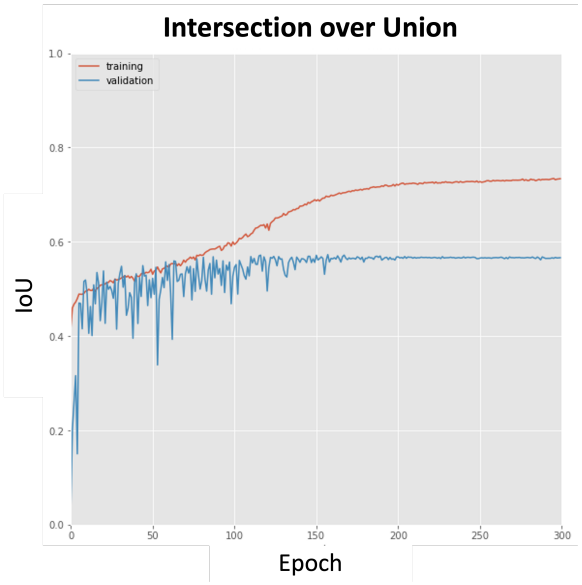


(a) OB Loss Function. (b) OB IoU.

Figure 20: Osteoblasts training loss function and Intersection over Union. In red are shown the training values, in blue the validation ones.

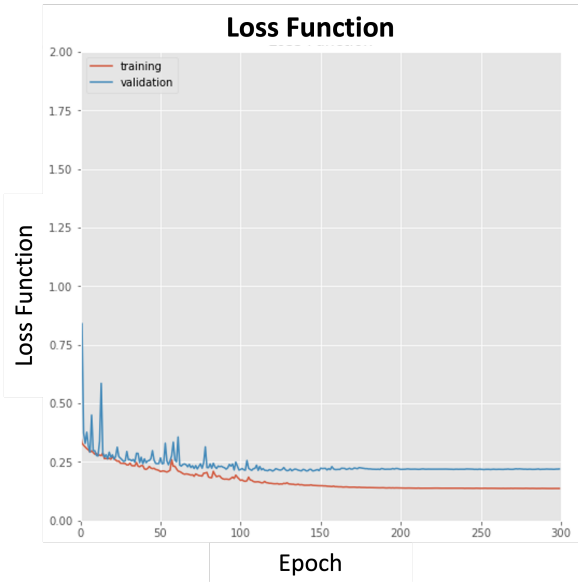


(a) OC Loss Function.

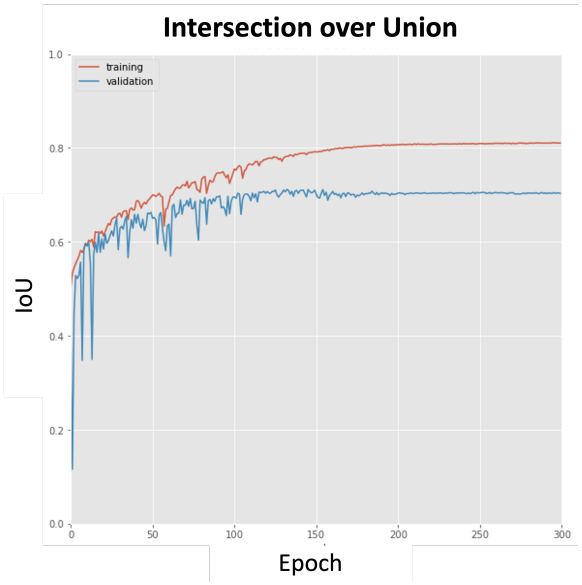


(b) OC IoU.

Figure 21: Osteoclasts training loss function and Intersection over Union

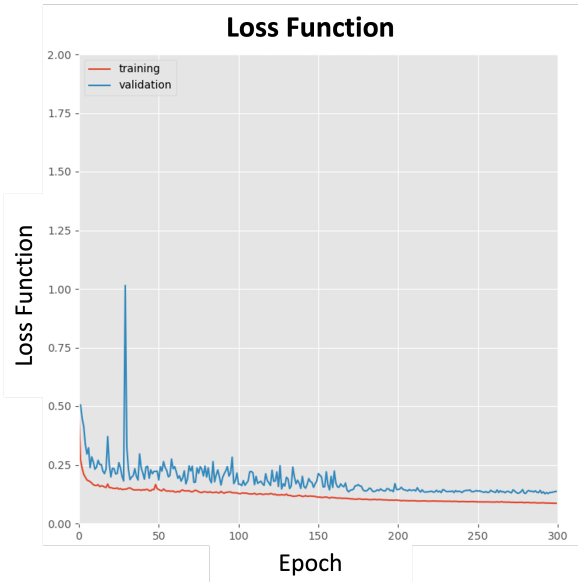


(a) Bone CM Loss Function.

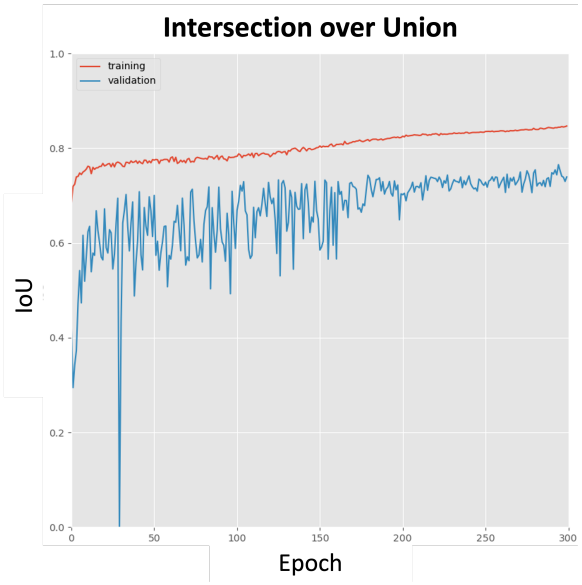


(b) Bone CM IoU.

Figure 22: Bone CM training loss function and Intersection over Union

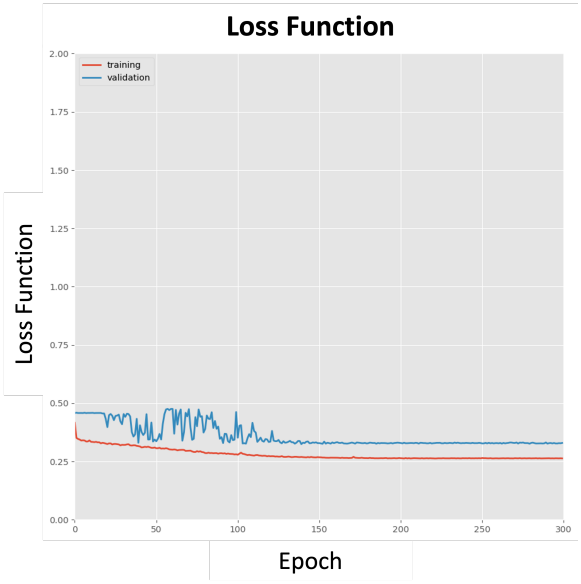


(a) Bone MPM Loss Function.

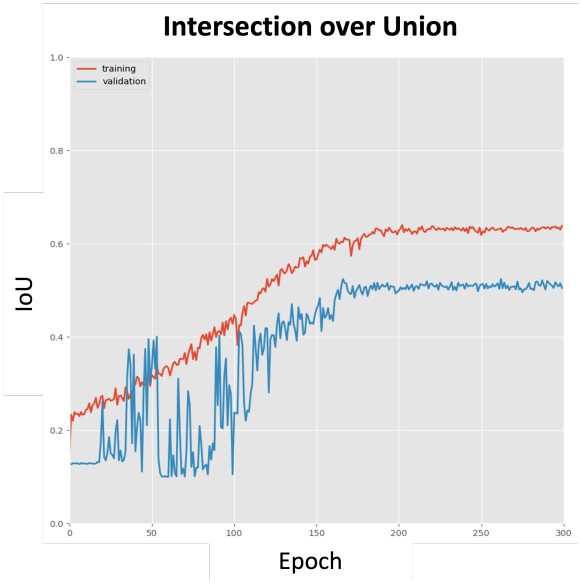


(b) Bone MPM IoU.

Figure 23: Bone MPM training loss function and Intersection over Union



(a) Vessels Loss Function



(b) Vessels IoU.

Figure 24: Vessels training and validation Loss Function and Intersection over Union

Models training were performed for 300 epochs for all cell types, as this setting showed the best trade-off between performance and training time. Both OBs' and OCs' training resulted in a higher training maximum IoU value (0.8345 and 0.7147, respectively) than the validation one (0.5517 and 0.5728, respectively) as the network was not able to learn the most complex features, given the limited training subset. The trainings for CM and MPM bone segmentation were characterized by a similar evolution over the epochs and reached comparable performances in terms of validation IoU (0.7118 and 0.7651, respectively). Bone trainings resulted in an overall better segmentation capability since the bone structures are more regular in the dataset. Vessels' model training was already performed in a previous work [45] and we limited our improvements in applying hyperparameter tuning and image patching, resulting in better performance in terms of validation IoU (0.42 of our work vs 0.34 of the previous work). To reach satisfactory training results, the cells' dataset were composed of both MPM and CM images (except for the vessels' dataset, composed only by CM images) that expressed different pixel intentensity values, increasing data variability and given the limited training subsets' size we performed Data Augmentation, as explained in Section 2.2. As explained in Section 2.4, different network architectures were

tested (custom U-Net, U-Net ++, ResNet and U-Net-ResNet34) to find the best performing model. Eventually, the best architecture was a U-Net with a ResNet34 as backbone with the validation results shown in Table 6, along with the epoch in correspondence of which these values were reached. Overall, even though not optimal, these metrics were graded as satisfying by biologists, who are the final users of this architecture.

The trainings' history (Figures 20, 21, 22, 23, 24), show fluctuations at the beginning of this procedure, due to the complex structures and information contained in the dataset. OB (Figure 20) and OC (Figure 21) IoU plots show a training curve that reaches higher values than the validation one. This behavior refers to a slight overfitting due to the limited dimensions of the dataset. The amount of information received by the architecture is not enough for the model to learn and generalize therefore, as the network keeps learning new features from the training images, these are not useful to better segment the validation images. The performance on the validation dataset remains stable, without getting worse. However, the trained models achieved satisfactory performances on the test dataset.

Cell Type	IoU	Epoch
OBs	0.5517	130
OCs	0.5728	158
Bone MPM	0.7651	294
Bone CM	0.7118	133
Vessels	0.4244	262

Table 6: Best IoU values over the validation dataset. The table shows also the epoch at which this value is reached during the training procedure.

Margin for improvement was found in applying the several models on patched or resized images, also depending on the image resolution and complexity. Therefore, in an effort to improve metrics, we implemented a function in our software that allows the user to choose the preprocessing procedure:

- patching 512x512: the image is patched with image2patch into 512x512 pixels smaller images with minimum pixel loss and overlap. This procedure feeds the model with zoomed-in portions of the same image; hence, more information on small structures but less spatial information. Moreover, it might lead to more accurate predictions as small regions are more visible but could be affected by FPs as also dark regions are highlighted and mistakenly segmented as cells.

- patching 1024x1024: this procedure works as patching 512x512, with larger patches (1024x1024 pixels). This step provides images that are less zoomed-in than the 512x512 ones, but give information about the entire image.

- resize 512x512: images are simply resized to meet the computational requirements of the models. This procedure provides the best information about the whole image but has the lowest resolution on small structures in the images. Predictions might result under-estimated with a higher incidence of FN.

3.2. Segmentation Model Evaluation

All the trained models were tested on new images and their performances were evaluated based on the differences between the predicted and manually segmented ground-truth masks. To quantify the differences between predicted and ground-truth masks, we used pixel-wise and cell-based metrics.

- **Pixel-wise metrics**:

$$IoU(U,V) = \frac{area\ of\ intersection}{area\ of\ union} \tag{9}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{10}$$

$$Precision = \frac{TP}{TP + FP} \tag{11}$$

$$Recall = \frac{TP}{TP + FN} \tag{12}$$

Where:
True Positive (TP): number of correctly classified foreground pixels (1). These are the pixels that the model correctly classifies as belonging to a cell structure.
False Positive (FP): number of misclassified foreground pixels.
True Negative (TN): number of correctly classified background pixels (0). These are the pixels that the model correctly assigns to the background (not belonging to any cell structure).
False Negative (FN): number of misclassified background pixels.

- **Cell-based metrics**. Based on the performance of the models in correctly classifying a region as a cell. A region is counted as a cell if all its pixels are 8-connected and if its area is larger than a threshold (10 pixels).
    - **Number of cells**. The number of correctly classified cells depends on the distance between the predicted and the ground truth cells. If this distance is lower than a threshold (15 pixels), the predicted region is counted as correctly classified as a cell. Otherwise, an error is counted. In other words, a cell is counted as correctly predicted if its centroid falls within 15 pixels from the centroid of the ground truth cell.
    - **Bone contour coverage**. Computed as the amount of OBs or OCs that lay on the bone contour (the region of interest for our study), and it is compared to the ground truth bone contour. This metric gives a more realistic idea on the accuracy and the reliability of our models since it does not consider non-interesting regions (as the bone marrow itself) where there could be prediction errors due to image acquisition artifacts and do not depend on the segmentation performance.

These metrics allow for models' performance evaluation and understanding, and future improvement directions.

Automatic Quantification of Osteoblasts. OBs are responsible for the deposition of new bone and their presence in the bone marrow is highly deregulated by bone metastases' effect. In control samples, OBs are spread all over the contour of the bone, covering almost all the perimeter of it. Therefore, the quantification and analysis of single OBs is impossible, hence we calculated their bone contour coverage, as explained in Section 2.5. In Table 7 are shown the results of the pixel-wise metrics for the OBs evaluation dataset. In Table 8 the results of the segmentation on bone coverage are shown for the ground truth and the predicted masks. The error is calculated as:

$$Error = \frac{|GroundTruth - Predicted|}{GroundTruth}\% \tag{13}$$

Image	Precision	Accuracy	Recall	IoU
Composite BPO 9-16-22	0.8246	0.9909	0.8831	0.7435
Composite S1 BPO 9-16-22	0.8274	0.9769	0.7983	0.6843
RAD223-AA-Pred S1 Composite	0.7106	0.944	0.8777	0.6466
4 weeks 3	0.8009	0.9910	0.6733	0.5768
Fused All Channels 2	0.6639	0.9858	0.8229	0.5809
Total	0.7659	0.9777	0.799	0.6375

Table 7: Pixel-wise metrics results of the OB segmentation model over the evaluation dataset.

Automatic Quantification of Osteoclasts. OCs are responsible for the digestion of old bone and their presence and activity plays an important role in osteolytic bone metastases. The number of OCs varies under the influence of tumors or therapies. OCs, differently from OBs, are characterized by irregular shapes and, in control samples, few single cells lay on the bone contour. To evaluate the performance of the segmentation model, we checked also the number of cells. Table 9 shows the results on the pixel-wise metrics. In Table 10 and Table 11 are shown the results for the Cell-Based metrics after a denoising procedure of the predicted masks. Bright points in the images might be segmented as OC, therefore there is an over-estimation of the OC number for the majority of the images. To overcome this limitation and provide more accurate results, we calculated the OC bone coverage that does not take into account cells that are far from the bone contour and therefore are not of interest.

Image	Ground Truth	Predicted	Error (%)
Composite BPO 9-16-22	93.725	92.078	1.757
Composite S1 BPO 9-16-22	77.688	75.907	2.293
RAD223-AA-Pred S1 Composite	89.319	84.841	5.013
4 weeks 3	59.359	58.024	2.249
Fused All Channels 2	21.146	21.892	3.528
Total			2.968

Table 8: OB bone coverage in the evaluation images. 4 weeks 3 and Fused All Channels 2 are characterized by low bone coverage because the acquired OB channel was dark and affected by noise and both manually and automatically generated masks were hard to be computed in the dark areas.

Image	Precision	Accuracy	Recall	IoU
Composite BPO 9-16-22	0.7927	0.9954	0.777	0.6457
Composite S1 BPO 9-16-22	0.6644	0.9896	0.7712	0.5550
RAD223-AA-Pred S1 Composite	0.7436	0.9951	0.7592	0.6016
4 weeks 3	0.8321	0.9927	0.8725	0.7419
Fused All Channels 2	0.8702	0.9982	0.8541	0.7576
Total	0.7806	0.9942	0.8068	0.66036

Table 9: Pixel-wise evaluation metrics results of the OC segmentation model over the evaluation dataset.

Image	Ground Truth	Predicted	Error (%)
Composite BPO 9-16-22	28.793	30.821	7.043
Composite S1 BPO 9-16-22	23.169	22.882	1.239
RAD223-AA-Pred S1 Composite	11.607	9.144	21.22
4 weeks 3	44.505	46.439	4.346
Fused All Channels 2	6.651	3.774	43.257
Total			15.42

Table 10: OC bone coverage in the evaluation dataset.

Image	Ground Truth	Predicted	Error (%)
Composite BPO 9-16-22	114	101	12.871
Composite S1 BPO 9-16-22	110	129	14.729
RAD223-AA-Pred S1 Composite	61	72	15.278
4 weeks 3	57	67	14.925
Fused All Channels 2	84	98	14.286
Total			14.418

Table 11: Number of OC counted in the evaluation dataset.

Automatic Quantification of Bone As explained in section 2.1 the bone dataset was split in MPM and CM images because the CM images do not have a proper bone channel. In CM images, in correspondence of bone structures, there is absence of signal, while in MPM images the bone channel is the second-harmonic generated signal. Therefore, we needed to split the dataset and train 2 different models. However, the performances on the training set are comparable, as shown in Figures 23b, 22b and Table 6. Table 12 shows the segmentation results for MPM and CM segmentation models on the evaluation datasets. We adopted only pixel-wise metrics since coverage and number of cells is useless to understand the performance on bone structure predictions.

Dataset	Precision	Accuracy	Recall	IoU
CM	0.9208	0.9463	0.8847	0.8215
MPM	0.9899	0.9745	0.9107	0.9019
Total	0.9485	0.9575	0.8950	0.85366

Table 12: Pixel-wise evaluation metrics results over CM and MPM Bone dataset with one model for each dataset. We adopted 3 CM images and 2 MPM images.

Automatic Quantification of Vessels Vessels analysis was only possible on CM images because the vessel segmentation model was trained on CM images. Tables 13 and 14 show the pixel-wise and cell-based segmentation results. A future development for this application would be generating a model specific for the segmentation of MPM vessels images.

Dataset	Precision	Accuracy	Recall	IoU
Composite BPO 9-16-22	0.7906	0.9408	0.7389	0.618
Composite S1 BPO 9-16-22	0.8108	0.9506	0.8451	0.6409
RAD223-AA-Pred S1 Composite	0.6865	0.9375	0.7203	0.542
Total	0.7626	0.9429	0.7681	0.6003

Table 13: Vessels pixel-wise evaluation metrics results.

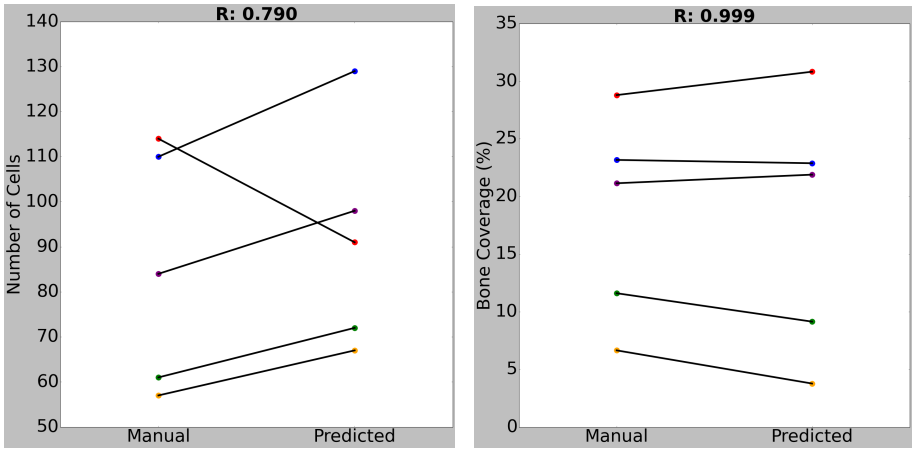
Image	Ground Truth	Predicted	Error (%)
Composite BPO 9-16-22	91	115	26.374
Composite S1 BPO 9-16-22	72	74	2.778
RAD223-AA-Pred S1 Composite	43	48	11.628
Total			13.593

Table 14: Number of Vessels counted in the evaluation dataset.

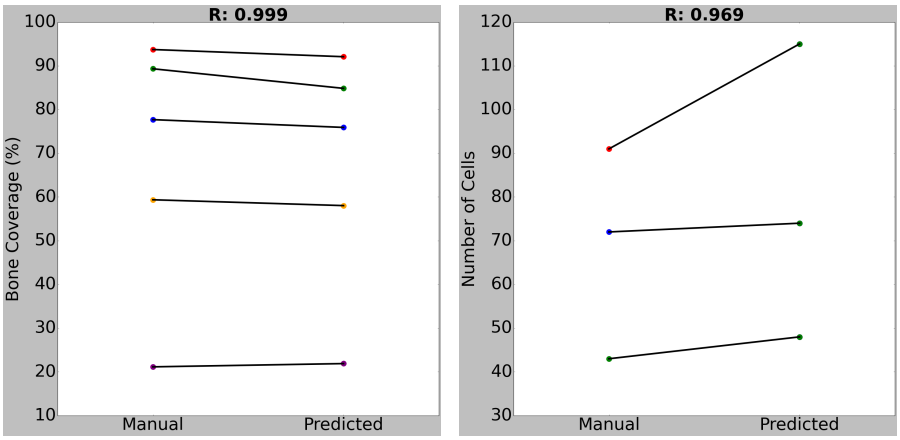
Figure 25 compares the predicted (25c) with the manually segmented (25b) images. The result is visually valid, also given the original image in 25a. Pearson coefficient (R) calculated between the predicted and the manual masks and shown in Figure 26. R = 0.790 for the number of OCs (26a), R = 0.999 for the OCs (26b) and OBs (26c) bone coverage and R = 0.969 for the number of vessels (26d). The coefficients are calculated over 5 independent CM and MPM images (3 for the vessels, CM images only). These results show strong correlation between the predictions and the ground-truth confirming the accuracy of the analysis in terms of number of cells and bone coverage.

(a) OC original image. (b) OC manual mask. (c) OC predicted mask.

Figure 25: OC original image (25a), manual (25b) and automatic (25c) segmentation from the test dataset.



(a) OC number. (b) OC coverage.



(c) OB coverage. (d) Vessel number.

Figure 26: Pearson correlation between manual and predicted outputs for: (26a) number of osteoclasts, (26b) osteoclasts bone coverage, (26c) osteobalsts bone coverage and (26d) number of vessels.

The overall computational time for each image, including the pre-processing pipeline (same applied for training and validation images), model prediction and postprocessing is in the order of seconds, providing a significant time saving compared to manual procedures.

3.3. CellHunter

CellHunter is a user-friendly software that incorporates DL models for image analysis. It comes with a graphical user interface (GUI) consisting of four main pages: "Home", "Add Image", "Results", and "Predicted Image". The "Home" page provides the user with instructions on how to use the tool, as shown in Figure 27. The "Add Image" page allows the user to upload an image for analysis, to select the channels in the image, and to choose a pre-processing method. The pre-processing options include patch sizes of 512x512 or 1024x1024 pixels, or resizing to 512 pixels. If the image is a CM image, the user must specify this to ensure appropriate pre-processing is applied. CellHunter also allows the user to save predicted masks generated by the DL models, and to modify them with ImageJ, and then re-upload them. In this case, the user can select an option to skip the pre-processing step. Figure 28 provides a visual explanation of this page. The "Results" page displays extracted parameters of interest in a table format that can be saved as an ".xlsx" file, as shown in Figure 29. Finally, the "Predicted Image" page displays the predicted masks after processing by the DL models. Figure 30 shows an example of this page.

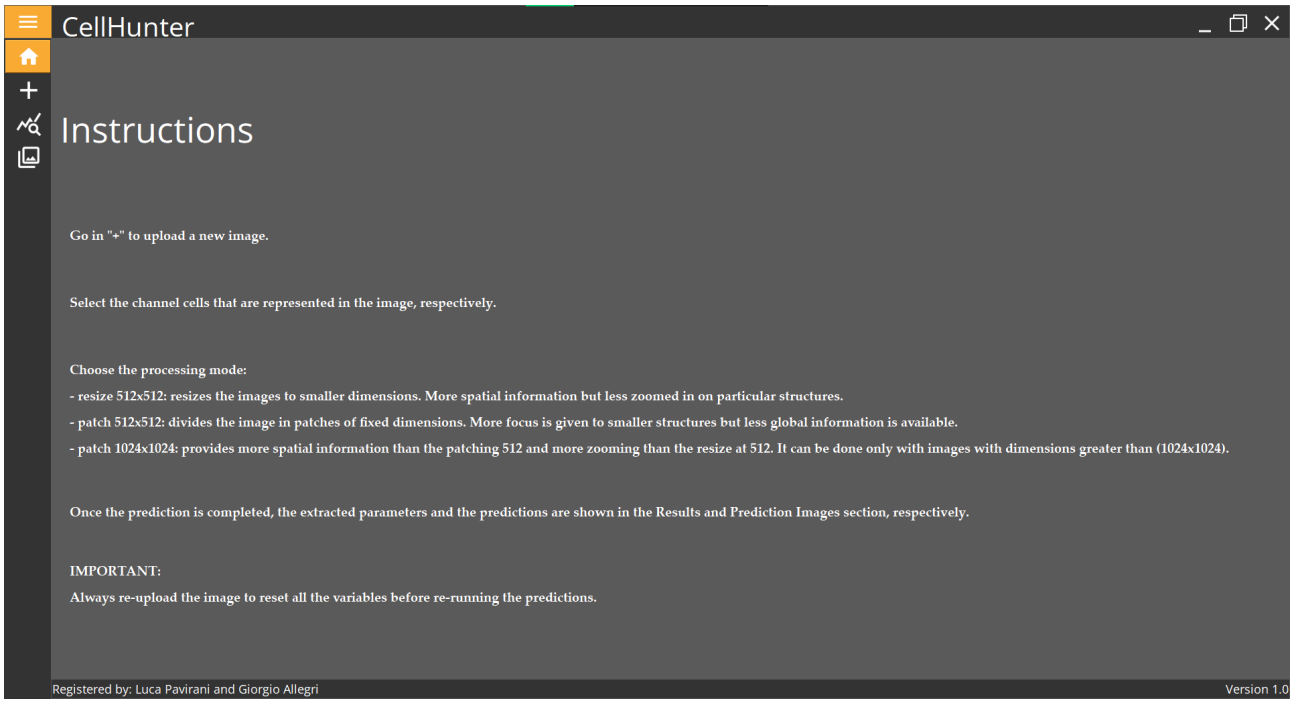


Figure 27: CellHunter: Home section.

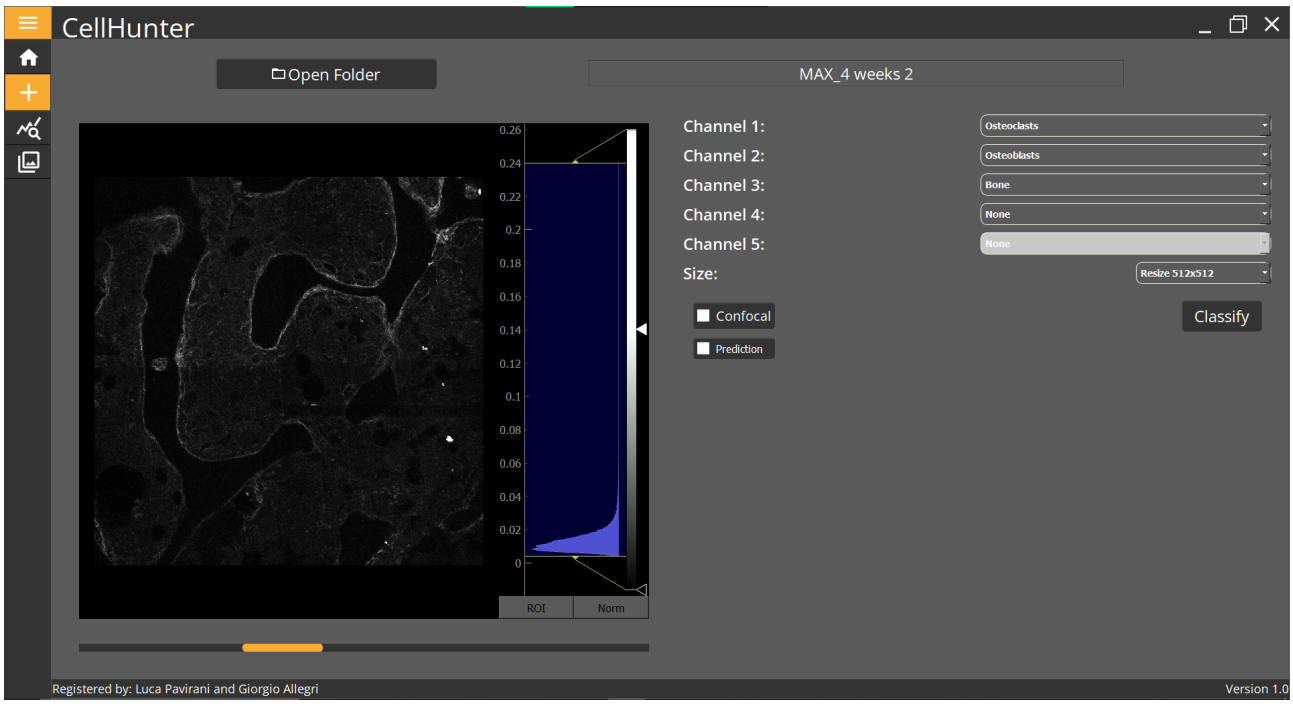
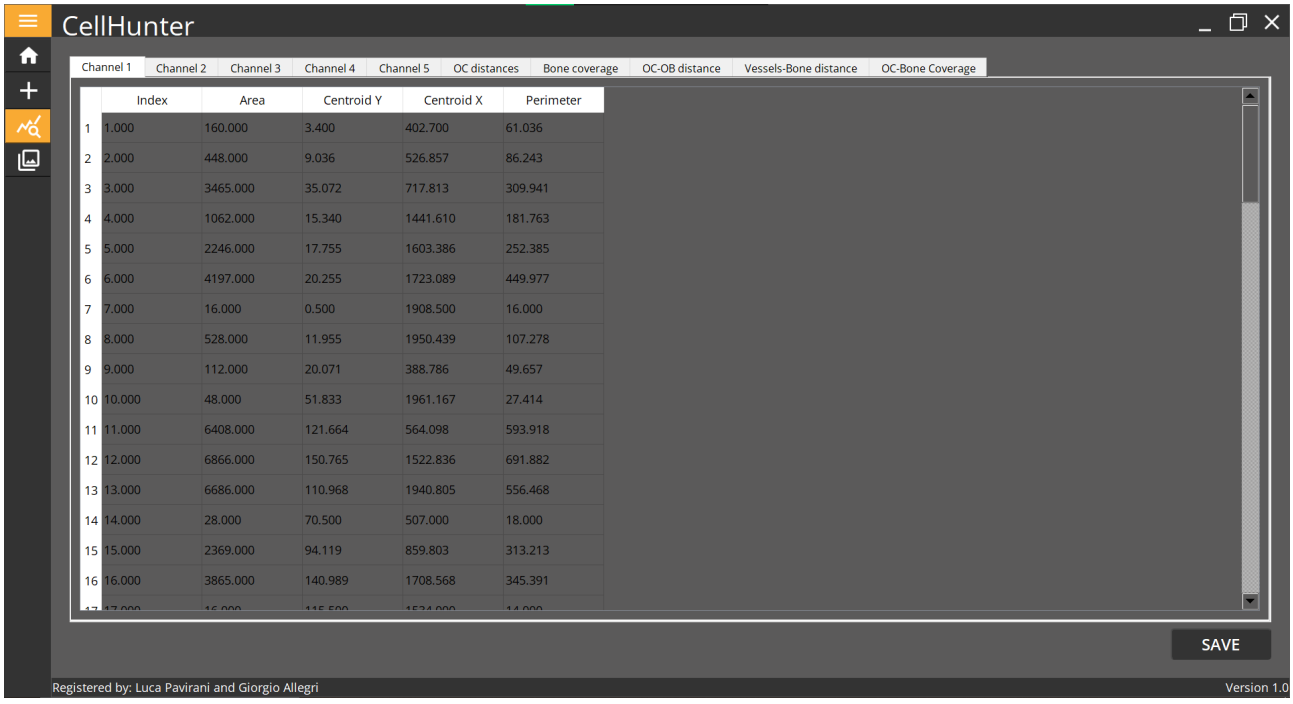


Figure 28: CellHunter: Add Image section.



	Index	Area	Centroid Y	Centroid X	Perimeter
1	1.000	160.000	3.400	402.700	61.036
2	2.000	448.000	9.036	526.857	86.243
3	3.000	3465.000	35.072	717.813	309.941
4	4.000	1062.000	15.340	1441.610	181.763
5	5.000	2246.000	17.755	1603.386	252.385
6	6.000	4197.000	20.255	1723.089	449.977
7	7.000	16.000	0.500	1908.500	16.000
8	8.000	528.000	11.955	1950.439	107.278
9	9.000	112.000	20.071	388.786	49.657
10	10.000	48.000	51.833	1961.167	27.414
11	11.000	6408.000	121.664	564.098	593.918
12	12.000	6866.000	150.765	1522.836	691.882
13	13.000	6686.000	110.968	1940.805	556.468
14	14.000	28.000	70.500	507.000	18.000
15	15.000	2369.000	94.119	859.803	313.213
16	16.000	3865.000	140.989	1708.568	345.391

Figure 29: CellHunter: Results section.

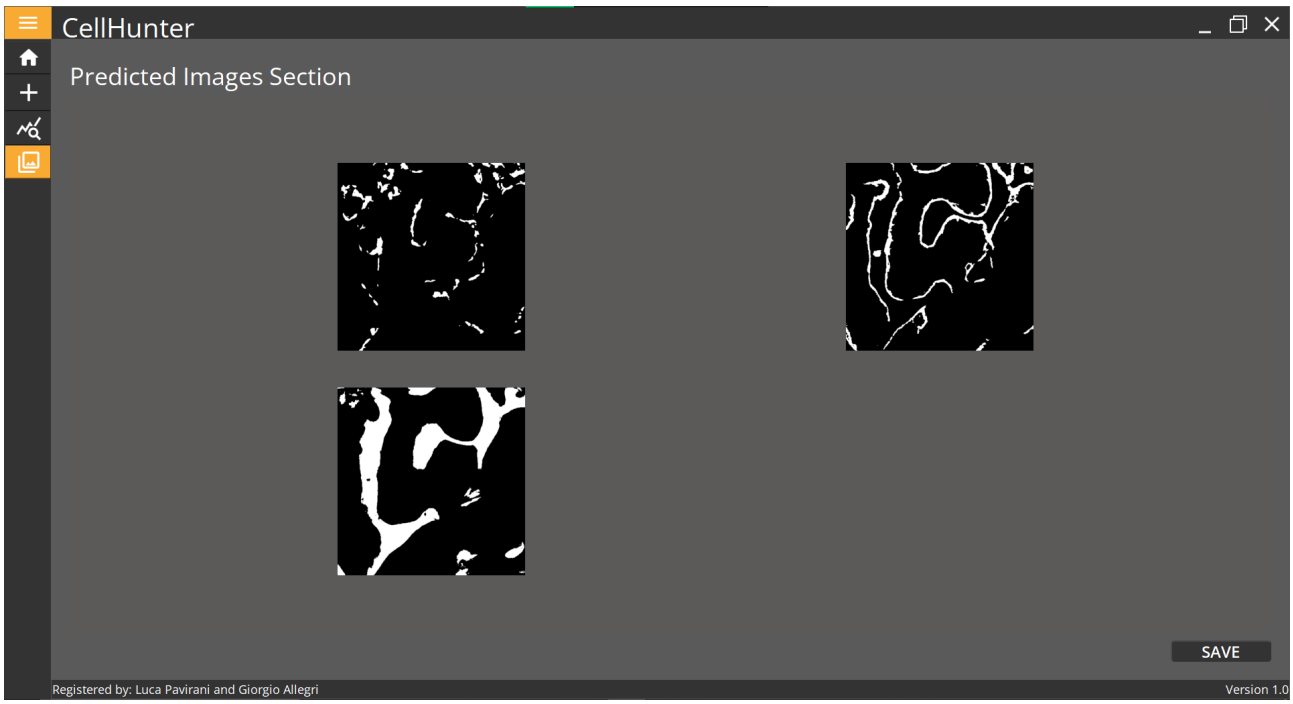


Figure 30: CellHunter: Predicted Image section.

4. Conclusion

In this study, we developed and implemented a tool to automate the analysis of MPM and CM images in the context of bone metastases. Our application employs DL techniques to perform semantic segmentation and extraction of measurements of interest, enabling biologists to obtain key features from microscopy images quickly and accurately. The application extract features such as the OB and OC bone coverage, cell size, area and relative position, blood vessel number, dimension, and distance from the bone contour. We validated the performance of our models through statistical metrics such as IoU, precision, recall, accuracy and biologically relevant measurements as the number of cells and the bone contour coverage. To increase the usability of our tool, we developed an executable software equipped with a Python-based GUI . This makes it easy for biologists without technical backgrounds to use our application. Furthermore, we developed a website where users can download our software for Windows and Linux operative systems.

As current limitations, restricted labeled dataset size constrains the performance of our DL models that require a bigger amount of data to perform more accurately. MPM and CM generate different microscopy images in terms of resolution, brightness, and contrast. This leads to a high variability within the dataset that is more complicated to be properly learned by the DL algorithms. To overcome this limitation a larger dataset is needed for both image types.

As future perspectives, we plan to expand the variability and availability of data to improve the performance of our models. We also aim to improve the accuracy and coherence of the labels used in our research, as well as explore the use of ground-truth images for evaluation. To increase the generalizability of our models, we plan to allow users to upload their own images for re-training through our website. Additionally, we plan to make our software available for other operative systems.

Overall, our application represents a significant contribution to the field of biomedical image analysis and cancer pre-clinical research. By automating the process of image segmentation and feature extraction, we have enabled biologists to save time and increase the reproducibility of their research.

References

[1] H. Schatten, "Brief overview of prostate cancer statistics, grading, diagnosis and treatment strategies," *Cell & molecular biology of prostate cancer: updates, insights and new frontiers*, pp. 1–14, 2018.

[2] J. Sturge, M. P. Caley, and J. Waxman, "Bone metastasis in prostate cancer: emerging therapeutic strategies," *Nature reviews Clinical oncology*, vol. 8, no. 6, p. 357, 2011.

[3] A. Jemal, R. Siegel, J. Xu, and E. Ward, "Cancer statistics, 2010," *CA: a cancer journal for clinicians*, vol. 60, no. 5, pp. 277–300, 2010.

[4] F. Macedo, K. Ladeira, F. Pinho, N. Saraiva, N. Bonito, L. Pinto, and F. Gonçalves, "Bone metastases: an overview," *Oncology reviews*, vol. 11, no. 1, 2017.

[5] D. J. Hadjidakis and I. I. Androulakis, "Bone remodeling," *Annals of the New York academy of sciences*, vol. 1092, no. 1, pp. 385–396, 2006.

[6] T. Ibrahim, E. Flamini, L. Mercatali, E. Sacanna, P. Serra, and D. Amadori, "Pathogenesis of osteoblastic bone metastases from prostate cancer," *Cancer: Interdisciplinary International Journal of the American Cancer Society*, vol. 116, no. 6, pp. 1406–1418, 2010.

[7] A. H. Jinnah, B. C. Zacks, C. U. Gwam, and B. A. Kerr, "Emerging and established models of bone metastasis," *Cancers*, vol. 10, no. 6, p. 176, 2018.

[8] F. Saad, D. M. Gleason, R. Murray, S. Tchekmedyian, P. Venner, L. Lacombe, J. L. Chin, J. J. Vinholes, J. A. Goas, and B. Chen, "A randomized, placebo-controlled trial of zoledronic acid in patients with hormone-refractory metastatic prostate carcinoma," *Journal of the National Cancer Institute*, vol. 94, no. 19, pp. 1458–1468, 2002.

[9] K. Fizazi, M. Carducci, M. Smith, R. Damião, J. Brown, L. Karsh, P. Milecki, N. Shore, M. Rader, H. Wang *et al.*, "Denosumab versus zoledronic acid for treatment of bone metastases in men with castration-resistant prostate cancer: a randomised, double-blind study," *The Lancet*, vol. 377, no. 9768, pp. 813–822, 2011.

[10] O. Sartor, A. J. Armstrong, C. Ahaghotu, D. G. McLeod, M. R. Cooperberg, D. F. Penson, P. W. Kantoff, N. J. Vogelzang, A. Hussain, C. M. Pieczonka *et al.*, "Survival of african-american and caucasian men after sipuleucel-t immunotherapy: outcomes from the proceed registry," *Prostate cancer and prostatic diseases*, vol. 23, no. 3, pp. 517–526, 2020.

[11] I. F. Tannock, R. De Wit, W. R. Berry, J. Horti, A. Pluzanska, K. N. Chi, S. Oudard, C. Théodore, N. D. James, I. Turesson *et al.*, "Docetaxel plus prednisone or mitoxantrone plus prednisone for advanced prostate cancer," *New England Journal of Medicine*, vol. 351, no. 15, pp. 1502–1512, 2004.

[12] J. S. De Bono, S. Oudard, M. Ozguroglu, S. Hansen, J.-P. Machiels, I. Kocak, G. Gravis, I. Bodrogi, M. J. Mackenzie, L. Shen *et al.*, "Prednisone plus cabazitaxel or mitoxantrone for metastatic castration-resistant prostate cancer progressing after docetaxel treatment: a randomised open-label trial," *The Lancet*, vol. 376, no. 9747, pp. 1147–1154, 2010.

[13] S. Lutz, L. Berk, E. Chang, E. Chow, C. Hahn, P. Hoskin, D. Howell, A. Konski, L. Kachnic, S. Lo *et al.*, "Palliative radiotherapy for bone metastases: an astro evidence-based guideline," *International Journal of Radiation Oncology* Biology* Physics*, vol. 79, no. 4, pp. 965–976, 2011.

[14] C. a. Parker, S. Nilsson, D. Heinrich, S. I. Helle, J. O'sullivan, S. D. Fosså, A. Chodacki, P. Wiechno, J. Logue, M. Seke *et al.*, "Alpha emitter radium-223 and survival in metastatic prostate cancer," *New England Journal of Medicine*, vol. 369, no. 3, pp. 213–223, 2013.

[15] T. M. Beer, E. D. Kwon, C. G. Drake, K. Fizazi, C. Logothetis, G. Gravis, V. Ganju, J. Polikoff, F. Saad, P. Humanski *et al.*, "Randomized, double-blind, phase iii trial of ipilimumab versus placebo in asymptomatic or minimally symptomatic patients with metastatic chemotherapy-naive castration-resistant prostate cancer," *J Clin Oncol*, vol. 35, no. 1, pp. 40–47, 2017.

[16] E. Dondossola, S. Alexander, B. M. Holzapfel, S. Filippini, M. W. Starbuck, R. M. Hoffman, N. Navone, E. M. De-Juan-Pardo, C. J. Logothetis, D. W. Hutmacher *et al.*, "Intravital microscopy of osteolytic progression and therapy response of cancer lesions in the bone," *Science translational medicine*, vol. 10, no. 452, p. eaao5726, 2018.

[17] K. N. Weilbaecher, T. A. Guise, and L. K. McCauley, "Cancer to bone: a fatal attraction," *Nature Reviews Cancer*, vol. 11, no. 6, pp. 411–425, 2011.

[18] F. Xing, Y. Xie, H. Su, F. Liu, and L. Yang, "Deep learning in microscopy image analysis: A survey," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 10, pp. 4550–4568, 2017.

[19] D. Shen, G. Wu, and H.-I. Suk, "Deep learning in medical image analysis," *Annual review of biomedical engineering*, vol. 19, pp. 221–248, 2017.

[20] M. Sarti, M. Parlani, L. Diaz-Gomez, A. G. Mikos, P. Cerveri, S. Casarin, and E. Dondossola, "Deep learning for automated analysis of cellular and extracellular components of the foreign body response in multiphoton microscopy images," *Frontiers in Bioengineering and Biotechnology*, vol. 9, p. 1497, 2022.

[21] N. F. Greenwald, G. Miller, E. Moen, A. Kong, A. Kagel, T. Dougherty, C. C. Fullaway, B. J. McIntosh, K. X. Leow, M. S. Schwartz *et al.*, "Whole-cell segmentation of tissue images with human-level performance using large-scale data annotation and deep learning," *Nature biotechnology*, vol. 40, no. 4, pp. 555–565, 2022.

[22] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. Van Der Laak, B. Van Ginneken, and C. I. Sánchez, "A survey on deep learning in medical image analysis," *Medical image analysis*, vol. 42, pp. 60–88, 2017.

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

[24] C. A. Schneider, W. S. Rasband, and K. W. Eliceiri, "Nih image to imagej: 25 years of image analysis," *Nature methods*, vol. 9, no. 7, pp. 671–675, 2012.

[25] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE transactions on systems, man, and cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[26] A. M. Reza, "Realization of the contrast limited adaptive histogram equalization (clahe) for real-time image enhancement," *Journal of VLSI signal processing systems for signal, image and video technology*, vol. 38, pp. 35–44, 2004.

[27] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*. Springer, 2015, pp. 234–241.

[28] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[29] W. Wu. patchify. [Online]. Available: https://github.com/dovahcrow/patchify.py

[30] G. Allegri and L. Pavirani, "image2patch," https://github.com/LucaPavirani/image2patch.py, 2022.

[31] A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, and A. A. Kalinin, "Albumentations: fast and flexible image augmentations," *Information*, vol. 11, no. 2, p. 125, 2020.

[32] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[33] F. Chollet *et al.* (2015) Keras. [Online]. Available: https://github.com/fchollet/keras

[34] P. Yakubovskiy. (2019) Segmentation models. [Online]. Available: https://github.com/qubvel/segmentation_models

[35] T. O'Malley, E. Bursztein, J. Long, F. Chollet, H. Jin, L. Invernizzi *et al.*, "Kerastuner," https://github.com/keras-team/keras-tuner, 2019.

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

[37] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014. [Online]. Available: https://arxiv.org/abs/1412.6980

[38] S. Jadon, "A survey of loss functions for semantic segmentation," in *2020 IEEE conference on computational intelligence in bioinformatics and computational biology (CIBCB)*. IEEE, 2020, pp. 1–7.

[39] Z. Zhou, M. M. Rahman Siddiquee, N. Tajbakhsh, and J. Liang, "Unet++: A nested u-net architecture for medical image segmentation," in *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support: 4th International Workshop, DLMIA 2018, and 8th International Workshop, ML-CDS 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 20, 2018, Proceedings 4*. Springer, 2018, pp. 3–11.

[40] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[41] J. Le'Clerc Arrastia, N. Heilenkötter, D. Otero Baguer, L. Hauberg-Lotte, T. Boskamp, S. Hetzer, N. Duschner, J. Schaller, and P. Maass, "Deeply supervised unet for semantic segmentation to assist dermatopathological assessment of basal cell carcinoma," *Journal of imaging*, vol. 7, no. 4, p. 71, 2021.

[42] "Adobe illustrator," [Computer software], Adobe Systems, San Jose, CA, 2021. [Online]. Available: https://www.adobe.com/products/illustrator.html

[43] "Qt designer," [Computer software], The Qt Company, Oslo, Norway, 2021. [Online]. Available: https://www.qt.io/product/designer

[44] R. C. Limited, "PyQt5 v5.15.6 reference guide," [Computer software], 2021, accessed: February 23, 2023. [Online]. Available: https://www.riverbankcomputing.com/software/pyqt/

[45] G. Alessandrelli, "Deep learning and image processing techniques to automate bone metastasis image analysis," Master's thesis, Politecnico di Milano, 2021.

A. Acronyms

PCa	Prostate Cancer
OB	Osteoblast
OC	Osteoclast
DL	Deep Learning
GUI	Graphic User Interface
MPM	Multiphoton Microscope
CM	Confocal Microscope
Alk P	Alkaline Phosphatase
TRAP	Tartate-Resistant Phosphatase
SHG	Second Harmonic Generation
DAPI	4',6-diamidino-2-phenylindole
NN	Neural Network
FCN	Fully Convolutional Network
CNN	Convolutional Neural Network
ML	Machine Learning
IoU	Intersection over Union
FP	False Positives
FN	False Negatives
TP	True Positives
TN	True Negatives
TL	Tversky Loss
TI	Tversky Index

Abstract in lingua italiana

Il Cancro alla Prostata é una malattia comune (12.5%) e mortale tra gli uomini, e le metastasi ne sono la principale causa di mortalitá. Le metastasi ossee causano gravi problemi di salute e riducono drasticamente la prospettiva di vita a 5 anni da 100% per il tumore primario a 30% per il tumore metastatico. In questo contesto, la ricerca pre-clinica fa forte affidamento sulla analisi di immagini di microscopia, attivitá che é attualmemte svolta manualmente dai biologi. L'automazione di questa procedura é fondamentale data la sua natura dispendiosa in termini di tempo, la bassa riproducibilitá e la propensione all'errore umano. Si ipotizza che l'applicazione di algoritmi di *Deep Learning* all'analisi di immagini di microscopia porterá al superamento delle limitazioni legate ai metodi manuali attuali, ottimizzando la ricerca pre-clinica. Di conseguenza, abbiamo sviluppato una architettura basata su reti neurali per eseguire segmentazione semantica di strutture cellulari di interesse come osteoblasti, osteoclasti, osso e vasi da immagini di microscopia di metastasi ossee di cancro alla prostata. La segmentazione é poi seguita da una fase di estrazione ed elaborazione di parametri di interesse a partire dalle immagini segmentate, con conseguente organizzazione in database. Nonostante la disponibilitá limitata di immagini, abbiamo ottenuto metriche soddisfacenti in termini di *Intersection over Union* (IoU). Il modello relativo alla segmentazione degli osteoclasti ha raggiunto un valore di IoU pari a 0.73 e 0.57 in addestramento e validazione, rispettivamente. Per quanto riguarda la segmentazione degli osteoblasti, é stato raggiunto IoU di 0.84 in addestramento e 0.55 in validazione, mentre, per la segmentazione dell'osso, le prestazioni su immagini di microscopio a multifotone (0.79 in addestramento e 0.71 in validazione) e confocale (0.81 in addestramento e 0.71 in validazione) sono comparabili. Infine, é stato sviluppato il modello per la segmentazione dei vasi, che ha raggiunto 0.64 in fase di addestramento e 0.42 in validazione. I biologi hanno testato i modelli, e hanno confermato l'accuratezza della segmentazione e l'importanza biologica del nostro sistema. Con questa struttura, abbiamo fornito ai biologi uno strumento automatizzato ed affidabile, che riduce il tempo necessario per l'analisi di immagini da alcune ore a minuti, ottimizzando la *pipeline* di elaborazione delle immagini. Sviluppi futuri si concentreranno sull'espansione del dataset disponibile e sul conseguente miglioramento dell'accuratezza nella segmentazione e sullo sviluppo di un software eseguibile anche per altri sistemi operativi.

Parole chiave: Metastasi Ossee, Cancro alla Prostata, Cellule Stromali, Deep Learning, Rete Neurale Convoluzionale, U-Net, Image2patch, Segmentazione Semantica, Interfaccia Grafica, Analisi di Immagini

Acknowledgements

We are extremely grateful to Prof. Pietro Cerveri, from the Department of Electronics, Information and Bioengineering, Politecnico di Milano, who gave us the opportunity to discover and deepen our interest and knowledge in this field.
We would also like to express our deepest appreciation to Prof. Stefano Casarin, from the Immunology and Transplant Science Center at Houston Methodist Research Institute, and Prof. Eleonora Dondossola, from the GU Medical Oncology Department at MD Anderson Cancer Center, who guided and followed us from the beginning of the project.
We are also grateful to Luca Marsilio, from the Department of Electronics, Information and Bioengineering, Politecnico di Milano, for his valuable help and ideas throughout the development of the work.

Ringraziamenti

Desidero esprimere la mia sincera gratitudine a tutti coloro che mi hanno sostenuto durante il percorso di laurea magistrale.
Innanzitutto, desidero ringraziare il mio professore Pietro Cerveri per avermi offerto la possibilità di lavorare a questo progetto di ricerca e per avermi supportato con la sua esperienza e competenza. Un ringraziamento speciale va ai miei correlatori Stefano Casarin ed Eleonora Dondossola, che mi hanno accompagnato nella fase di ricerca e mi hanno fornito preziosi consigli e suggerimenti per migliorare il mio lavoro. Non posso dimenticare di ringraziare i miei genitori, i miei fratelli e la mia famiglia, per il sostegno morale e affettivo che mi hanno sempre fornito. Grazie per aver creduto in me e per avermi spronato a perseguire i miei obiettivi. Siete stati sempre presenti in ogni momento, e non avrei potuto fare tutto questo senza di voi. Voglio inoltre ringraziare tutti gli amici, per la loro vicinanza e il loro supporto in ogni momento, anche quando eravamo lontani. Siete stati un'importante fonte di conforto e incoraggiamento. Infine, desidero ringraziare tutte le persone che ho incontrato durante questo percorso, per le conversazioni, le sfide e le esperienze che mi hanno arricchito personalmente e professionalmente.
Grazie di cuore a tutti.