Automated Deep Learning Models for the Analysis of Biological Microscopy Images

Haja, Asmaa

DOI:
10.33612/diss.886191331

2 DETAILED OVERVIEW OF THE BACKGROUND

Deep learning is widely applied in the biological field for object detection, segmentation and classification. It becomes more successful as it performs certain image classification tasks with accuracy comparable to that of a human and requires little human input in the form of manually designed features or arbitrarily chosen thresholds [39]. This section reviews several recent works related to the introduced challenges in chapter 1. Each of the following chapters will have a related work section corresponding to the particular approached issue. Additionally, the data is introduced in section 7.2, as well as the computing tasks are defined in section 2.3. Finally, the utilized deep learning models are introduced in section 2.4.

2.1 RESEARCH IN MICROSCOPIC IMAGE ANALYSIS

A finite number of existing studies in the broader literature have examined object detection, segmentation and classification for microscopic images using deep learning (DL). Greenwald et al. [40] constructed a dataset with more than 1 million manually labelled cells for training Mesmer, a segmentation model. It was demonstrated that this approach (Mesmer) is more accurate than previous methods that achieved human-level performance and can be generalized to the full diversity of tissue types. To reduce the number of manual annotations, Moshkov et al. [41] used various augmentation techniques to increase both the training and test datasets. Then they performed the prediction both on the original and on the augmented versions of the image, followed by merging the predictions. This approach is called test-time augmentation (TTA). Finally, they incorporated the TTA prediction method into two major segmentation approaches utilized in the single-cell analysis of microscopy images. Both studies, though, need to have or create a large number of annotated examples to be used as input to their DL models. [42] presented a method for detecting nano-particles and determination of their shapes and sizes simultaneously with deep learning. Multiple-output convolutional neural networks (MO-CNN) are used in the proposed technique, and they generate two outputs: the first is the detection output, which provides the locations of the particles, and the second is the segmentation output, which provides the boundaries of the nano-particles. Using the segmentation output as input, the modified Hough algorithm calculates the particle sizes in the final

state. Their approach works mainly for round-shaped nano-particles in electro microscopy (EM) images. Again, MO-CNN also has the limitation that it needs a dataset, marked at the pixel level, for training. These research examples show the struggle in searching for different solutions as microscopic data is not rich in the number of images, more specifically, annotated images. The **self-supervised** concept demonstrated in part I is one of the possible current solutions to this problem.

A limited number of existing studies in the broader literature have examined the detection and segmentation of **overlapping** objects in microscopic images. One method employed by [43] is to segment near-circular objects, such as cell nuclei from fluorescence microscopy images, by combining the "gas of near circles" active contour model to their method. However, active contour is not a reliable method since it is based on the pixel intensities, very slow, and the final result heavily depends on the initialization as well as the contour evolution energy functional [44]. [45] detects overlapping instances based on optimizing a tree-structured discrete graphical model. Their model requires weakly annotated (dotted) images from the user repressing the centers of the objects. In [46], two fully convolutional regression networks are used to estimate the number of cells in the image as well as segment their borders. In order to overcome the cell-overlapping problem they used a larger receptive field (filters) [47]. To tackle the overlapping nuclei problem, [48] created a framework that combines the convolutional neural networks with the marker-controlled watershed. One drawback of their models is the usage of the watershed in the post-processing part, which limits the generalization ability of their approach. As it can be seen the research on the overlapping objects in microscopic images in general, for the organoid dataset in particular, is still limited. Yet, accurate segmenting of the overlapping organoids' morphology is critical. A proposed is explored in part II discusses this issue.

Tools were also developed to aid in the automation of microscopic image **segmentation**. For instance, YeastNet [49] was designed to improve the accuracy of identifying individual S. cerevisiae cells from bright-field microscopy images. Their model was trained using a manually labelled dataset and is based on the U-Net semantic segmentation architecture. Recently, a new version of the CellProfiler [50] was deployed. Cellprofiler is a user-friendly software designed for biologists to perform a series of image-processing modules, such as segmentation. As a software application, it has a major potential disadvantage in that CellProfiler runs on local machine with a GPU. In this case, it needs local space and memory. Additionally, sharing processed data might not be easily accomplished. Therefore, an **end-to-end system** is needed, where the user should not be worried about his/her local space and memory and can easily share the data with other scientists from the same department.

As can be seen, most DL models adapted to the microscopic images to detect, segment or classify images require a sufficient amount of labelled data. Part I deals with this topic by employing the **self-supervised learning** technique and delve into the complexity of the pretext task. In part II, the segmentation of **overlapping organoid** images are presented and discussed. Part III propose an **end-to-end system** for the tasks performed on diverse microscopic images that can be used without installation.

2.2 MICROSCOPIC IMAGES USED IN THIS STUDY

In this thesis, two datasets are exploited to tackle the different issues mentioned earlier: (1) Organoid culture images obtained from the University Medical Center Groningen, and (2) Yeast strains images from the NOP1pr-GFP-SWAT library provided by the Weizmann Institute of Science. Figure 1.1 illustrates an abstract of the relationship between the organoid to cell to cell-compartments, also called organelle. In this section, the raw data is introduced.

2.2.1 Organoids

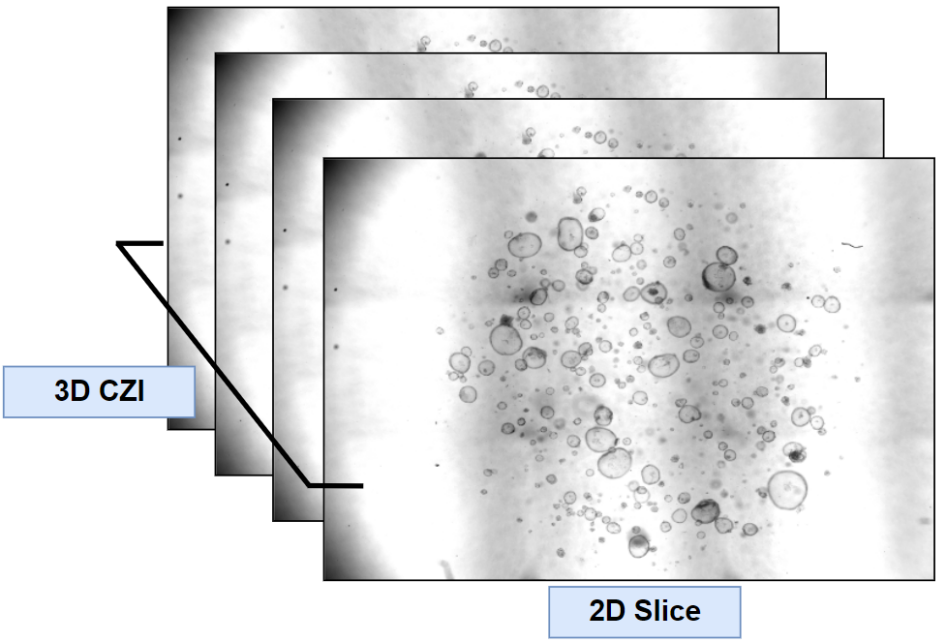

Figure 2.1: An example of a CZI: A 3D image made up of 2D slices, called stacks, at different depths of the organoid culture.

The organoids were developed under two distinctive conditions: (1) Organoids were grown where the essential amino acids were present (Control group), and (2) organoids were grown with the absence of amino acids (Starvation group). The organoid cultures were examined at five different time points using a specialized microscope at 24-hour intervals. A total of 10 3D CZI images, each consisting of 14 2D stacks (slices), are studied, yet, an average of 4 stacks per image were evaluated as the lower and higher stacks do not contain much information. Figure 2.1 presents an example of the 3D CZI images. The size of each stack image is 3828x2870 pixels. Those 2D stack images were used to generate a large number of smaller images that can be used as a training dataset. This has been done by moving a square sliding window, of size 636x636 pixels, over each 2D slice and generating cropped subsections of the latter. This procedure created a large number of 2D images, which have all been resized to 320x320 pixels in order to reduce the number of needed training parameters and increase the trainability of the model while not losing any significant details. The following chapters utilizing this dataset provide more information related to the specific questions to be addressed.

2.2.2 Cells

Images of yeast-cell genome-wide library of $\sim$ 5,500 strains carrying the SWAp-Tag (SWAT) NOP1promoter-GFP module at the N terminus of proteins are studied [51]. Yeast cells, are round to long [52], possess ultrastructural features typical of other eukaryotic cells, with the presence of membrane-bound organelles (Figure 2.2) [53]. It should be noted that the real microscopic image looks much less clear

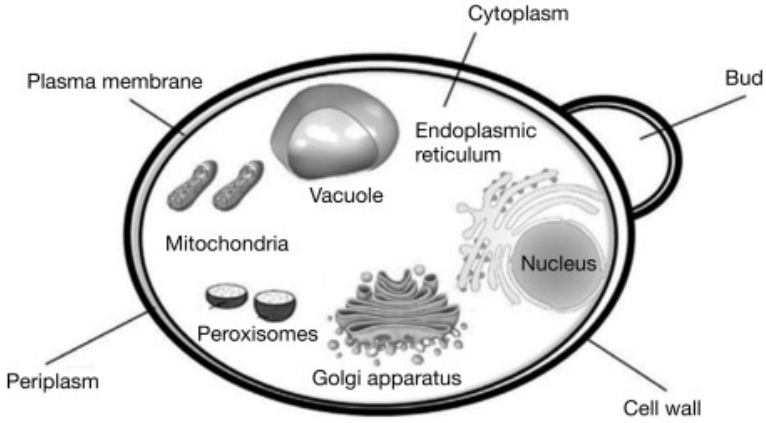

Figure 2.2: Yeast cell and its main organelles [53].

than this stylized diagram. The dataset contains images from 16 plates. Each well has images taken at 3 positions. Each well contains many cells of a yeast strain

producing a different protein N-terminally tagged with the green fluorescent protein (GFP) technique. Each image is captured in a brightfield channel as well as by fluorescence microscopy to detect the GFP-tagged protein, and is annotated by both the name of the protein and the name of the structure or compartment that it is inhabiting. Images' size is 1344x1024 pixels.

2.2.3 Organelles

The NOP1promoter-GFP dataset contains multiple subcellular localizations. This fluorescent dataset contains bright-field and colored tagged organelles. Table 7.1 presents the number of unique images for each subcellular localization ordered by the highest count. Only the first highest unique count of more than 70 images are shown. Note that some images contain two proteins tagged with the

Table 2.1: Number of unique images for different organelles in the NOP1promoter-GFP dataset.

	Unique number of Images
cytosol	1566
nucleus	660
mitochondria	461
cytosol,nucleus	401
ER	376
punctate	313
cytosol,punctate	223
nucleolus	140
punctate,nucleus	129
ER,vacuole	111
nucleus,nucleolus	96
vacuole membrane	92
ER,punctate	81
vacuole	79

corresponding fluorescent colours. For instance, "cytosol,punctate" are images that contain both tagged cytosol and punctate cell compartments. Figure 2.3 show such an image by merging the bright-field and the GPF channels.

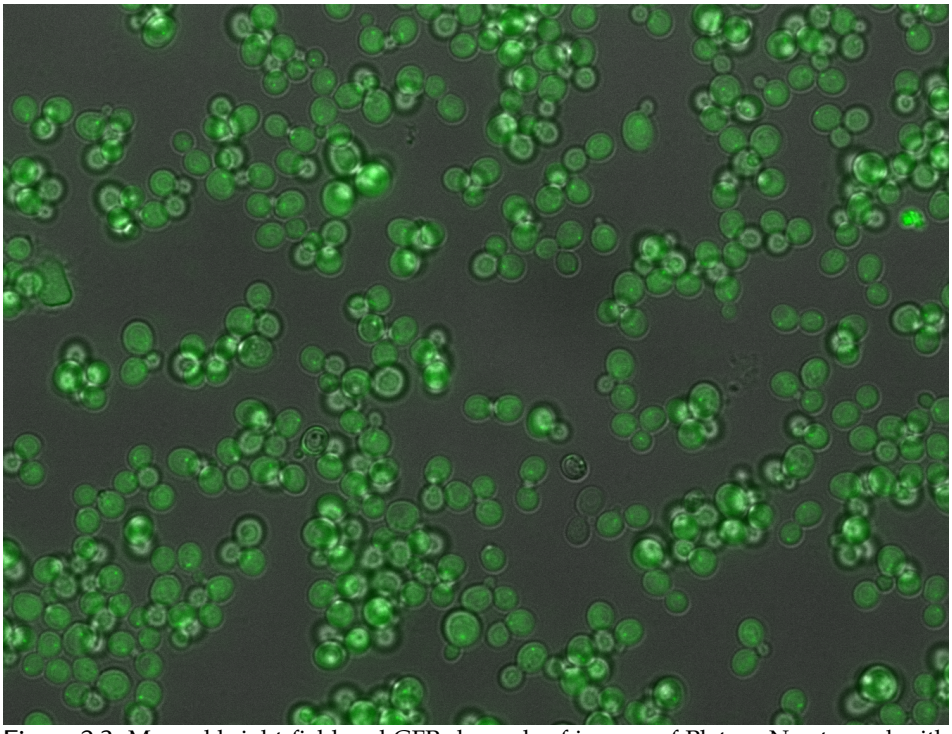

Figure 2.3: Merged bright-field and GFP channels of images of Plate 11N14 tagged with "cytosol,punctate".

2.3 COMPUTING TASKS

Object detection, segmentation and classification are the common basic tasks performed by any DL model. Supervised- and unsupervised learning are the most well-known techniques for training DL models. In supervised learning, a DL model learns patterns and relationships between input and output data. In this case, the output data, which is the target to be reached by the model, is known. Figure 2.4 explains the relationship between the input and output for the four previously mentioned basic tasks during the training step.

OBJECT DETECTION In object detection, the model identifies and locates the objects in the image. The models' input is an image, whilst the models' output is a prediction bounding box coordinates and the corresponding labels for each object. A post-processing step is applied to add the detected bounding box object to the predicted output image. A bounding box surrounds each detected object is shown in the first row of the Figure. A zoomed-in part of the detected object is shown on the right side of Figure 2.4.

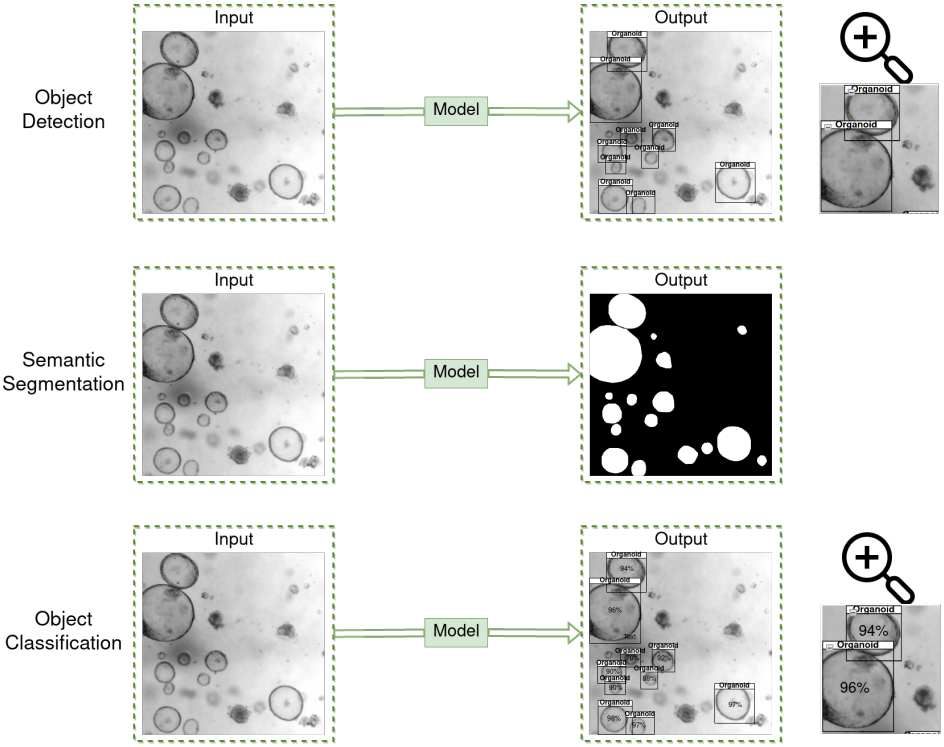

Figure 2.4: Deep learning tasks for training model via supervised technique.

SEMANTIC SEGMENTATION DL model performing a semantic segmentation task separates these objects from the background and from each other. Here, the model classifies each pixel as a background [coloured in black] or object [coloured in white]. This is the case for binary segmentation, as shown in Figure 2.4. For multi-class problems, the model predicts a pre-defined class for each object, which is used to colour the output image.

IMAGE CLASSIFICATION DL model trained to classify the complete images outputs a probability class vector indicating the probability of this image containing objects from a predefined set of possible categories/classes. The highest probability value is used to classify the images.

OBJECT CLASSIFICATION Object classification is a combination of image classification and object detection. Here, the class probability vector along with bounding box coordinates are computed for each object in the image and not for the complete image. A zoomed-in part of the classified object is shown on the right side of Figure 2.4.

Testing the trained DL model on an unseen dataset is required in order to assess its performance. For testing the model, the input is the ground truth image, and the output depends on the performed DL task.

On the other hand, the outputs for the unsupervised training models are unknown. Usually, such model clusters or group images together that look alike. Unsupervised learning is not used in this thesis because the goal is to guide the model to predict specific output and not to cluster images together.

Training DL models with supervised learning are only possible when a large number of annotated data is available. With regards to the segmentation task, the ground truth images are much more difficult to annotate than the classification images. It is necessary to sketch the target object's outline, which is a very laborious and time-consuming process. Thus, manually segmenting objects for each input image is almost impossible for a large amount of dataset. Self-supervised training can be used to solve this issue using two phases: (1) Training a model to solve a **pretext task**, and (2) adapting the trained model to handle the performed **main task**. Part I of this thesis deeply delves into the self-supervised topic of segmenting microscopic organoid images.

2.4 DEEP LEARNING MODELS

Deep learning models are frequently used for understanding and processing biological data as well as for extracting high-level abstract features. They outperform traditional models in terms of performance and interpretability [54]. The structure of most deep learning models is based on convolutional neural networks (CNN)s. In addition to the CNN as a base model, this section introduces the models employed in the following chapters.

2.4.1 Convolutional Neural Network

Convolutional neural networks (CNN)s are a type of artificial neural network (ANN) used most frequently in deep learning to analyze visual data. CNN consist of three elements: (1) Convolutional Layer, (2) Pooling, and (3) Flattening + Fully-Connected Neural Network.

CONVOLUTIONAL LAYER AND FEATURE REPRESENTATION The objective of a convolutional layer is to extract features that help in identifying the objects in the figure. A Kernel, typically called a filter of a 3x3 matrix, slides over the input image / feature maps from left to right and top to bottom and performs a dot product of its values with the sub-region values. A kernel is shown in the red

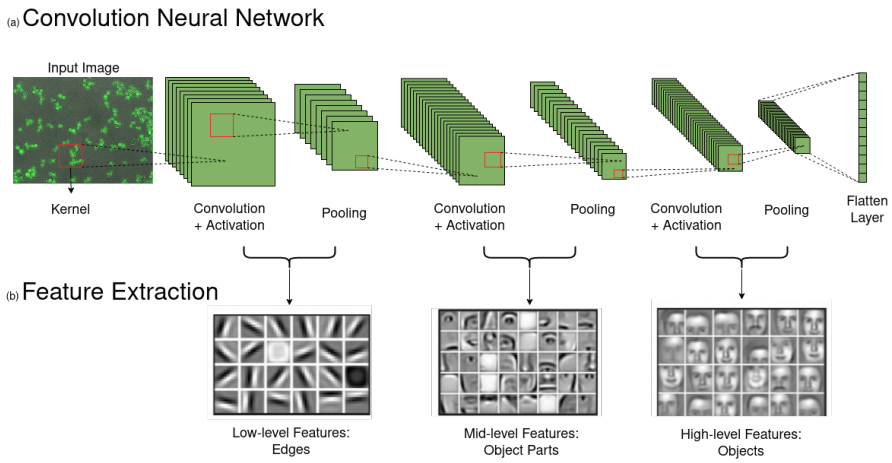

Figure 2.5: (Top) CNN architecture. (Bottom) feature extraction in stages, Source [55].

box of Figure 2.5. During this process, features will be extracted as shown in the bottom of Figure 2.5. In the beginning, low-level features are extracted, which represents edges, difference in pixels, etc. The mid-level features integrate the information (e.g. combine the edges) from the features before and can represent some object parts. The high-level features are able to represent the desired objects. To note that those features from a faces dataset are used as an illustration.

POOLING LAYER The more kernels are utilized, the more feature maps is collected. This has a limitation on the memory when training the CNN model. In order to decrease the number of computations in the network, the spatial size of the input image / feature maps is gradually reduced using the pooling layer. This is seen in Figure 2.5, where the number of the third dimension does not change from the previous layer, however, the first and second dimensions are halved as Max pooling is depicted.

FLATTENING The flattening layer is used to flatten all values from the last feature maps into a single 1D vector in order to perform one of the computation tasks described in section 2.3. Commonly, the CNNs' flattened layer is used for a computing bottleneck 'embedding' vector and as a step towards classification.

Although it is a powerful property of the CNN that it is translation invariant (important features can occur anywhere in the image) the subsequent limitation is that it fails to encode the position and orientation of objects, especially for small objects [56]. Another limitation of a deep CNN for classification is its requirement of a huge number of training samples [57], due to the large number of coefficients (weights) in a model. Since this thesis focuses on object segmentation and not

classification, a simple and basic CNN cannot be applied: In segmentation, the exact spatial location of input features needs to be known at the output level. For this purpose, a new generation of CNN models was developed, as described in the next section.

2.4.2 U-Net

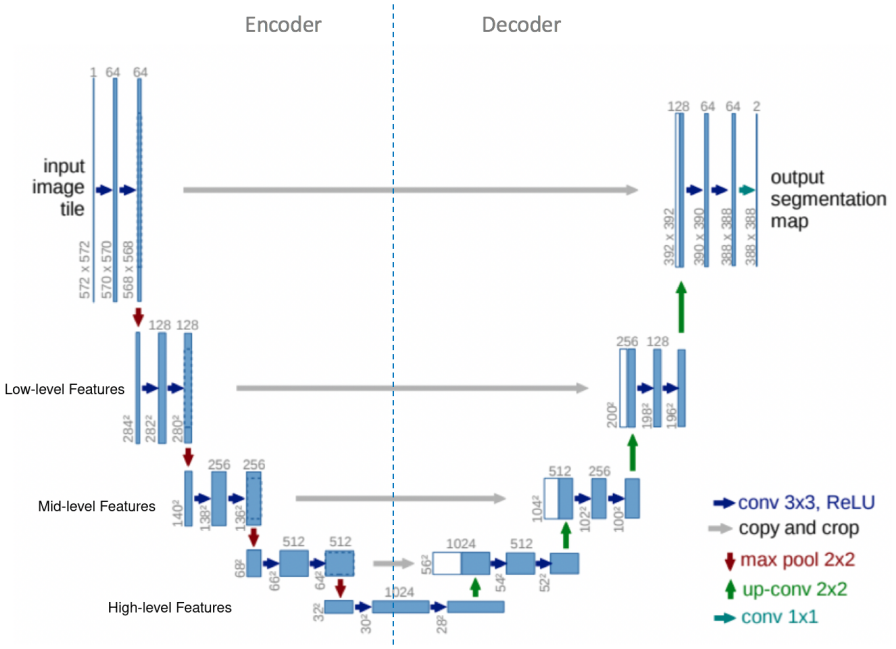

Figure 2.6: U-Net architecture. The encoder is on the left side, whilst the decoder is on the right side.

U-Nets are a type of Convolutional Neural Networks (CNN)s that were originally introduced for biological image segmentation [58]. They are comprised of an encoder-decoder architecture with skip connections that enable the network to retain fine-grained spatial information while also capturing global context. The U-Net consists of downsampling (encoder) layers and up-sampling (decoder) layers, which can be seen in Figure 2.6. The encoder encodes the image using various convolution and pooling layers over the input image, while the decoder decodes this information with the goal of reconstructing an output image that is exactly the same size as the input image. The low-, mid- and high-level features explained in section 2.4.1 are (suggestively) illustrated on the encoder path of the U-Net.

Despite its many noteworthy benefits, the U-Net model still has some drawbacks. For example, the model's structure is not flexible when trained on datasets of various image sizes, and the skip connection has not fully utilized the encoder block's features [59]. Another drawback is that U-Net-based models concentrate only on the last feature output of the convolution unit and forgetting the feature of the previous convolution in the node [60]. Owing to the popularity and success of U-Net, studies are working towards further improving the U-Nets' structure for medical image segmentation [59].

2.4.3 You Only Look Once

YOLO stands for "You Only Look Once". It is a deep learning algorithm that is mainly used for object detection and classification. As the name suggests, the model goes through the entire image only once. This makes YOLO a powerful and a fast algorithm, distinguishing it from other deep learning algorithms, which need two steps to detect and then classify individual objects in an image, namely by 1) finding regions of interest that suggest the location where individuals objects might be, and subsequently 2) classifying them.

YOLO was first introduced in [61], which is later named as YOLOv1. The general idea of YOLO is to divide the image into **S**x**S** grids, in which each grid cell is responsible for encapsulating an object of which the center is located within that grid (compare Figure 2.7) Additionally, each **S**x**S** grid cell is responsible

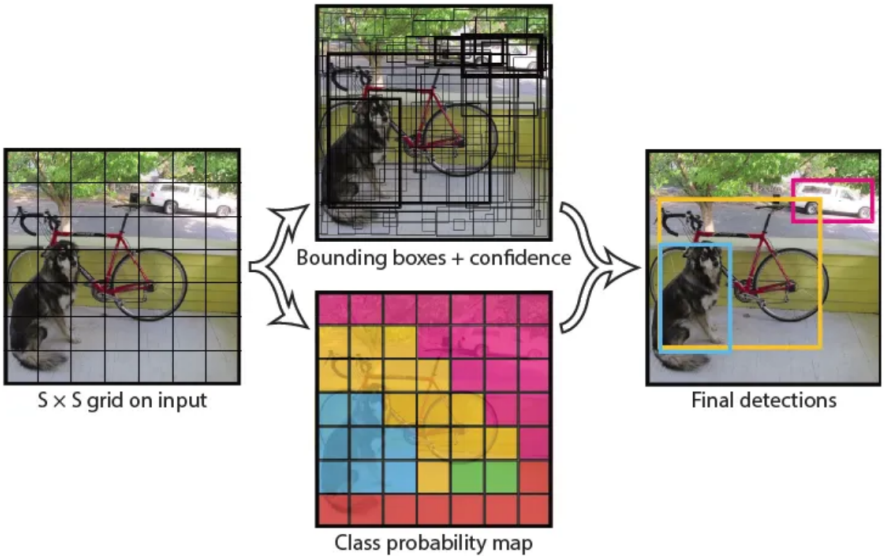

Figure 2.7: YOLOv1's Grid, BBox, Confidence scores and the class probability maps [61].

for detecting **k** Bounding Boxes [BBox]. Since not all BBoxs cells will contain an object, YOLO provides a confidence score to select the most likely BBox in a grid cell, containing an object. These scores vary between 0 and 1, and describe the probability of the existence of any pre-defined object. A confidence score equal to zero indicates a detection of background, whereas a high confidence score indicates a high probability of a pre-defined object being presented in the found BBox within the current grid cell. The confidence score is computed using the Intersection Over Union [IoU] between the ground truth BBoxs obtained from Mask-RCNN and the predicted BBoxs gained from YOLO. Moreover, YOLO produces (**x**, **y**, **width**, **height**) parameter values to indicate the location and dimension of the predicted BBoxs. The parameters **x** and **y** specify the x and y center location of the BBox relative to the grid cell, whereas **width** and **height** are the normalized dimension of the BBox with regards to the width and height of the entire image. It is to be noticed that **width** and **height** of a BBox can be bigger than the dimension of the defined grid cells. Finally, YOLO outputs **C** conditional class probabilities for each grid cell, where C is the number of classes/objects to detect. In total, YOLO outputs **S** x **S** x (**k** x **5** + **C**), where number 5 represents (x, y, width, height) and confidence score for each BBox. Despite YOLOv1's ability to simultaneously predicting BBoxs of different classes for an image in a short time, it has one main downside, which is precisely detecting and localizing small objects. This is due to the fact that multiple small objects can be presented in one grid cells, and YOLOv1 is able to detect only one object per grid cell.

In this thesis, the fourth version of YOLO is implemented. This version was developed by Bochkovskiy, Wang and Liao [62], since the previous authors wanted to retreat from the field of computer vision. The primary goal of their paper is to design a fast-operating object detector for production systems that is also optimized for parallel computations, and more importantly is that the training should be done on one single conventional GPU. To achieve this, the authors made a few changes to the object detector network architecture. In their paper, the authors compared different methods that are used for a one-stage detector, namely backbone, neck and head (also called dense prediction). CSP-Darknet53, which employs the Cross-Stage-Partial Network [CSPNet] strategy [63], is the optimal model to be used as a backbone for the detector network. The authors integrated the Spatial Pyramid Pooling [SPP] layer to the backbone by replacing the last pooling layer [64]. For the neck, the authors replaced the feature pyramid network [FPN] [65] used in YOLOv3 with Path aggregation network [PANet] [66]. They modified the output of PANet by replacing the addition operator of multiple feature maps with a concatenation operator. With regards to the head of the object detector network, the authors employ YOLOv3 as a head for YOLOv4. Furthermore, the authors made some design improvements, for example by modifying some existing methods: Spatial Attention Module [SAM], and Cross mini-batch Normalization [CmBN]; and introducing two new methods

for data augmentation: Mosaic and Self-Adversarial Training [SAT]. In summary, YOLOv4 - in comparison to other existing state-of-the-art models - is considered to be a significant upgrade to YOLOv3 in terms of speed, accuracy and performance. Not only that but it seems a good candidate to use for detecting small objects seeing all modifications that were added to YOLOv4. Therefore we consider it as the best starting point for addressing the where and what question, i.e., detection and classification, in microscopic images. Despite its success, one of YOLOv4 limitations is the inability to handle overlapping objects [67].

2.4.4 Loss-Function Design

The loss function is a mathematical instrument for assessing how effectively the deep learning model is able to produce desired outputs. The loss value will be high if the predictions are completely incorrect, whereas a low loss value indicates that the predictions are close to the expected known output. That is to say, the loss is responsible to steer the model towards being capable to accurately modelling the given input dataset. Mathematically, the loss function measures the difference between a prediction and the actual true value, e.g., as an L1 or L2 loss.

Based on the loss value, the models will update their weights during the back-propagation phase. This is mainly done by the optimizer, which is accountable for adjusting the model's parameters to minimize the loss function. The weights are updated in small steps, such that the model follows a path towards lower losses. This is called gradient descent, which is the most popular optimizer that is widely used [68]. Gradient descent can be realized by, first, computing the derivative of the loss function with respect to each of the parameters in the model, then updating those parameters proportional to the direction of the negative gradient. A limitation of gradient descent is that a single step size (learning rate) is used for all input variables. An adaptive moving estimation algorithm (Adam for short) is an extension to gradient descent that automatically adapts a learning rate for each input variable for the objective function [69].
In short, the loss function is considered to answer the "what to minimize" question, whilst the optimizer algorithm should answer the "how to minimize the loss" question. The most popular optimizer that is widely used is gradient descent [68] [70].

This thesis uses and explores various loss functions depending on the research question. Keeping in mind that each of the following chapters tackles a research question from a different perspective, the corresponding utilized loss functions will be introduced and explained in the corresponding chapters.

Part I

SPARSELY-LABELED BIOLOGICAL IMAGES