Politecnico di Torino

Corso di Laurea Magistrale in Ingegneria Biomedica
A.a. 2020/2021
Sessione di Laurea dicembre 2021

A hybrid deep learning framework for segmentation of crowded objects on medical images: application to digital pathology image analysis

Relatori:
Molinari Filippo
Salvi Massimo

Candidati:
Costa Luca

Index

List of figures

Abstract

Deep learning (DL) has brought great benefits to the whole medical imaging field, allowing for unprecedented performance in tasks that proved to be prohibitive to traditional image processing techniques. In particular, the introduction of Convolutional Neural Networks, with their great feature extraction and generalization capabilities, marked a great advance in segmentation problems, which are one of the most common and relevant challenges when dealing with medical images. Despite the various benefits that these kind of techniques carry, deep learning algorithms are still mostly seen as black-box systems, making it hard to understand the factors controlling their performance and to tune them without delving into architectural changes, which ultimately requires high expertise in the field. Recently, more and more researchers are hence focusing on trying to leverage on existing models by introducing additional pre/postprocessing steps to the pipeline, with the purpose of improving the baseline performance of the model. In this optic, this work aims to develop a hybrid framework, combining DL methods with traditional image processing techniques to segment medical images. In particular, semantic segmentation operated by CNNs usually shows very high pixel-level accuracy, but is prone to merging different objects that are close to each other or even intersecting. For this reason, the possibilities of such a framework are here explored on histopathological images. Moreover, Whole Slide Images (WSIs), a digital representation of a whole slide of tissue taken from a patient in a surgical environment during a biopsy, are becoming the primary source of observation for pathologists over direct observation of the tissue samples. Therefore, researchers are currently addressing the increasing need for automatic quantitative analysis tools that can lead to faster and more evidence-based diagnosis. The proposed method is built on a synergic combination of a deep segmentation network and an object detection model, to take advantage of the pixel-level segmentation accuracy of the former and of the higher spatial localization of whole instances of the latter. The information coming from both is then used to initialize and evolve Active Contours on the softmax probability map generated by the segmentation network and a repulsive interaction term is introduced to correctly handle the segmentation of those nuclei that were otherwise fused in the original segmentation, solely based on the network. The method is benchmarked on a standard dataset, specifically proposed for the evaluation of nuclear segmentation tasks, in terms of several pixel-level and object-level metrics and particular focus is put on evaluating the results in terms of Aggregated Jaccard Index, a comprehensive indicator for segmentation quality. The proposed approach shows statistically significant improvements over the performance baseline consisting of a common threshold-based segmentation of the softmax, suggesting the method's suitability for crowded images segmentation. Further work may include exploring the framework applicability and usefulness to other medical imaging domains: the inherent generality of the pipeline should however make the transfer relatively easy. Other possible improvements involve carrying a combined optimization of the two DL models to maximize performance returns and addressing potential scalability problems in the application of this framework to large amounts of data such as in entire WSI segmentations.

Introduction

1.1 Aim of the thesis

The application of digital processing techniques to images in the medical field have seen a constant growth in the recent years, mostly due to increasing computing capabilities, which enabled the development of more complex algorithms and their use in real-time application [1]. Nowadays, most of these techniques do not aim to have a direct diagnostic value, but configure themselves as a support to clinicians by addressing inherent problems such as repetitive manual tasks, inter-operator and intra-operator variability [2]. In particular, one of the research area that has recently gained a lot of interest is deep learning, which achieved state of the art performance in many medical imaging tasks [3]. The aim of this thesis is to develop and explore the potential benefits of a hybrid framework for medical image segmentation, combining some of the most recent deep learning algorithms with more traditional and established image processing techniques. More in depth, the developed method can be briefly summarised in the following steps:

- Training of a segmentation network and an object detection network on the same input data.
- Softmax probability map generation (as the segmentation network output) and centroid extraction (from the object detection bounding boxes).
- Initialization of non-parametric Active Contour models (commonly known as “snakes”) on the previously calculated centroids.
- Evolution of the active contours on the softmax.
- Final segmentation, obtained as the binary mask of each fully developed contour.

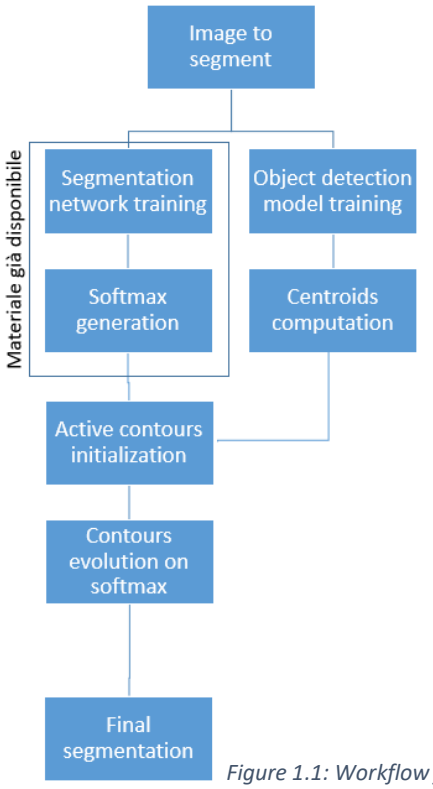


Figure 1.1: Workflow for the proposed framework

This approach should allow to take advantage of the outstanding generalization capabilities of deep learning, as well as the high shape adaptability of deformable models. Moreover, the combined segmentation and object detection networks can help overcoming the problem of fused segmentations of close objects. On these premises, this framework should therefore yield the highest benefit when used for the segmentation of images with a high density of close, possibly overlapping, objects. For this reason, the technique has been first developed and evaluated on digital histological images, even though its high generality should allow a reasonably simple transfer to any other medical imaging domain. This proposed approach follows the philosophy of an increasing number of recent studies, which are concentrating on increasing the baseline performance of classical deep learning techniques through the application of alternative post-processing techniques [4].

1.2 Histological imaging

Histology is a branch of biology that aims to study the microanatomy of biological specimens trough a microscope at different organization levels, ranging from the single cells to tissue and whole organ level [5]. Sometimes separate terms are used to indicate the study at each of those level (cytology for cells, histology for tissues and organology for organs), but more frequently the word histology is used to include them all.

1.2.1 Slide preparation and acquisition

Before a tissue specimen is ready for microscopic evaluation or image acquisition through dedicated digital scanners, it has to face a number of standard steps that ensure an optimal slide preparation [6]:

1. Fixation: through appropriate chemical fixatives, the tissue is locked in its current state. This makes it possible to preserve and maintain the structure of tissues and cells and to avoid autolysis and degradation of the sample with time. Moreover, fixation aids the subsequent process of cutting by hardening the tissue. The most widely employed agent for light microscopy is 10% neutral buffered formalin, used in a ratio of about 10:1 with the specimen volume.
2. Trimming: after choosing the portion of the sample that will be relevant for future investigation, the tissue is cut through a scalpel so that the resulting section can fit into a labelled tissue cassette.
3. Tissue processing: several steps to make the sample ready for sectioning in thin slices. First, the specimen is dehydrated (removal of water and fixation agent) by dipping it into increasing concentrations of alcohol. After cleaning through an organic solvent, the specimen is infiltrated with an embedding agent, most commonly molten paraffin wax, which provides a tough support matrix after solidification.
4. Sectioning: after a brief exposure to low temperatures to harden the waxed block, the specimen is ready to be sectioned in thin slices. The cuts are carried out by a microtome at an average thickness of 5 µm for light microscopy applications or even less for the electron counterpart.
5. Staining: at this stage most cells and other tissue structures of the specimen are transparent and show relatively low contrast when scanned through any microscope. Staining is hence

applied to enhance the contrast as well as to highlight structures of interest. There are several staining techniques available for use, but the typical general-purpose choice is the hematoxylin and eosin staining (H&E). Hematoxylin marks cell nuclei in blue, whereas eosin stains cytoplasm, extracellular matrix and different other structures in various shades of pink. Eventually, an optical glue is applied on top of each slide to protect the underlying specimen.

Thanks to current availability of quick and high-resolution whole-slide scanners, the prepared specimens are almost always digitized to whole-slide images (WSIs) and the anatomopathologists often analyse the samples directly on those stored digital images. The advantage in terms of space given by the storage of all samples in digital form is apparent, but as of today clinical histological laboratories are still required to conserve the physical slides for at least 10 years [7].

Figure 1.2: Motic™ EasyScan WSI scanner

Another direct consequence of using WSIs is that the operator is naturally enabled to use any available automatic algorithm or CAD at will during his diagnostic operations. Completely manual histological images analysis is, indeed, a very time consuming and potentially repetitive task which could benefit in several ways from the use of automated systems. For instance, roughly 80% of the biopsies analysed each year in the US are benign [8] and even in the case of pathologic slides, a vast portion of tissue can bring little diagnostic importance: an automatic identification of regions of interest can thus significantly reduce the time spent by operators reviewing healthy slides. Besides this time-saving aspect, researchers and pathologists agree on the importance of quantitative analysis in the diagnostic process: characteristics such as nuclei shape, spatial arrangement and size are correlated to tissue condition or sometimes can even help predict therapeutic response and they are all weighted during the educated decisions taken by pathologists [9][10].

1.2.2 Challenges for Image Processing

Despite all these benefits that digital image processing can bring to the histopathology field, there are several challenges which any algorithm dealing with histological images has to face [11]:

- Data density of histopatology images can be extremely high [12]: typical WSI scanners acquire images at up 40x magnification, resulting in 15000x15000 WSIs with a spatial resolution of 0,25 µm per pixel. Considering a 24 bits color depth, the total amount of data for a 1 mm^2 area is roughly 48 MB. Comparing these numbers to the average radiological image (CT or MRI scans), which is 512x512 pixels and color-coded on 2 bytes, it is clear that any automatic method working on this kind of images needs to be as efficient and optimized as possible to be able to run in a practical use case. Viewing software usually handles this

problematic by adopting a multi-resolution approach (pyramid representation): the fully magnified image is embedded in a single file with copies (typically three or four) of the same image, downsampled to lower magnification levels. At loading time, the software retrieves the portion of image requested by the user at the most suitable resolution level, depending on the zoom level relative to the whole image and taking into consideration that, at any time, the highest resolution that can actually be appreciated is limited by the screen resolution in use. Of course this approach results in an increased storage requirement, but it also makes the loading much faster at higher FOV values, as shown in the picture below.

- WSIs show a high inter-laboratory and inter-slide variability. In the first place, this is linked to the inherent heterogeneity of histological samples, which can be extracted from virtually any body part and contain a variety of different structures. This is one of the reason why deep learning, with its high generalization capabilities, has outperformed any traditional technique when it comes to histological classification and segmentation tasks. However, even when the same tissue type is considered, variations can be significant. This is mainly due to two factors: the previously described process of slides preparation, which can introduce variations between laboratories even when adopting the same protocols and staining, and digitization of the slide through different scanners [11]. The very same WSI acquired by two scanning equipment can show variations in both lightning and coloration.

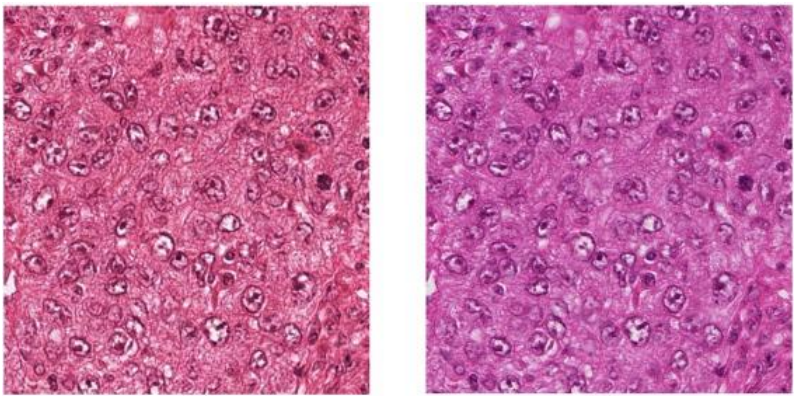


Figure 1.3: Same portion of H&E stained tissue acquired through two different WSI scanners

- Relevant information in histological images can be spread through different scales: depending on the diagnostic query, pathologists may be looking at single cells appearance, arrangement of groups of nuclei or higher-level structures like glands. An effective and complete CAD should therefore integrate information from all the needed levels, or combine single methods working at different scales.
- Lack of labelled data. While a good amount of publicly accessible unlabelled data is available, the lack of consistent training data represents a significant barrier to the development of performing and well-generalizing machine learning frameworks. Moreover, most labelled data available only contains whole-slide information regarding the condition of the sample (e.g. healthy/tumour) and is hence only useful for methods approaching classification tasks [13].

1.3 Introduction to Deep Learning

Deep learning is a subset of machine learning methods based on artificial neural networks, where information is passed and processed sequentially by a large number of layers [14]. One of the most relevant differences between traditional algorithms and machine learning techniques (and as such, DL) is the use they make of data they are applied on: while the former simply take data as an input and apply a set of rules (as complex as they might be), machine learning algorithms are designed to learn directly from data [15]. In other words, in traditional workflows data only represents the starting point of the entire pipeline, whereas it is the core element for any machine learning approach. As a direct consequence, the performance of ML implementations increases along with the quantity and quality of data available, as they get closer to be a representative sample for the specific problem considered. This concept is also linked to one of the first dissimilarities between general ML and DL: the graph below shows a qualitative estimation of the performance of both against the amount of data available [16]. By looking at the two curves, one can easily assert that DL potentially shows a much higher scalability than ML. In fact, even though ML performance may even be higher for small sized datasets, it eventually reaches a plateau, which results in negligible benefits for further increasing amounts of data; this does not happen to DL for which more data, paired with bigger (deeper) models, results in increased performance. This is perhaps the main reason why DL has only seen heavy adoption in the last decade, although the concepts behind it existed since the second half of the previous century: the sheer increase of publicly available digital data made it possible to build systems that fall in the right part of the graph, while the amount previously available could not exploit the full potential of DL [17].

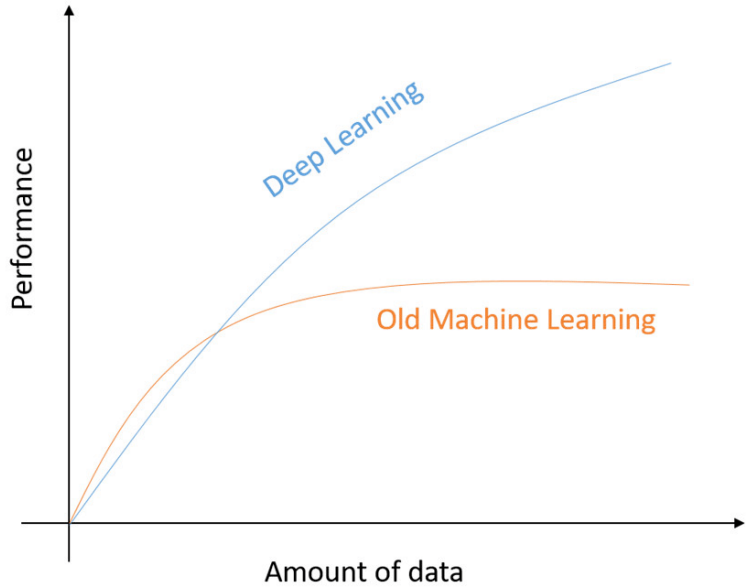


Figure 1.4: DL vs ML techniques scaling

Another import factor that sets DL apart from ML is the nature of the input data. ML algorithms usually rely on feature extraction, starting from the raw data, to craft specific features to be given as input to the system. On the other hand, DL is most often fed with raw data (eg. a full signal or image), as feature extraction is carried out internally by the network itself, with information processed by each layer into increasingly complex and high level features [14].

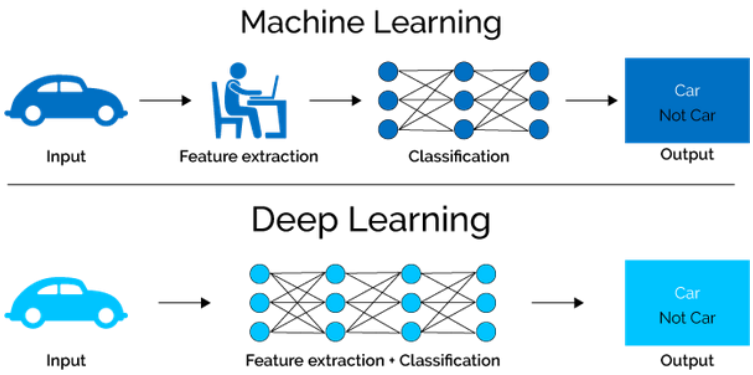


Figure 1.5: Differences in feature extraction for ML and DL methods

1.3.1 Artificial Neural Networks

Artificial Neural Networks are the core of most deep learning methods. As its name suggests, an ANN is a system that tries to mimic the inner working of the animal brain, which as a first approximation is made up of multiple neurons linked together by synapses [18].

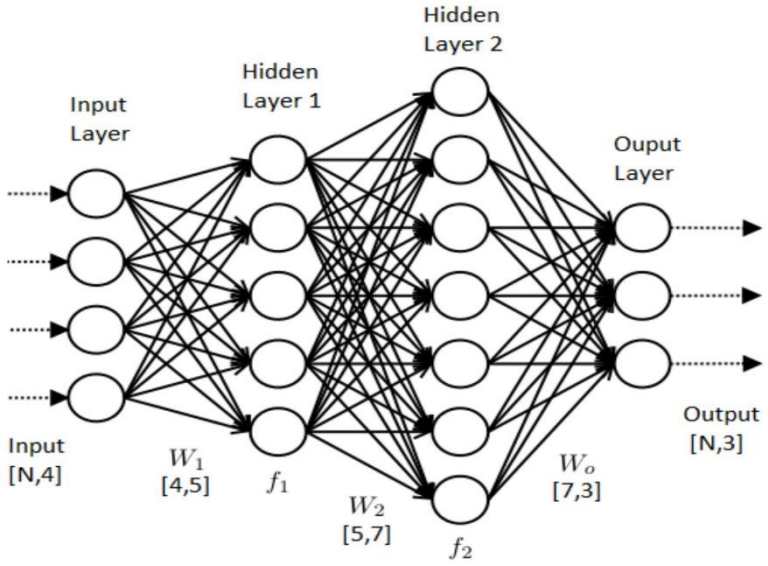


Figure 1.6: Typical Feed-Forward NN architecture

1.3.1.1 Neurons and activation functions

Indeed, the basic unit of every ANN is the artificial neuron, which in its most common form is simply composed by a linear regression model and an activation function. Given a set of inputs $\{x_1,x_2,\ldots,x_n\}$ (the input to the system or the outputs of a previous layer neurons), the output of the neuron z is:

$$z = w_1 * x_1 + w_2 * x_2 + \cdots + w_n * x_n + b = \sum_{i=1}^{n} w_i * x_i + b$$

where $\{w_1,w_2,\ldots,w_n\}$ are the weights of the input connections to the neuron (also called synapses to keep the biological analogy) and b is a constant bias term. In most cases, this result is then passed through a specific function to generate the final output a as: $a = f(z)$. The first and simplest

attempt to create and train a neural network, known as perceptron, actually consist of this single neuron block with a hard limiter activation function and dates back to 1958; it is hence only suitable for binary linear classification problems. There is a great number of activation functions employed in the literature [19], ranging from the identity function, that simply transfer the input to the output, to more complex non-linear ones; the table below shows some of the most common among them. The requisite for a generic function to be used as an activation is differentiability, as its derivatives are calculated during the training process. The main advantage of using an activation function that differs from the linear ones is that it makes the network more capable of learning highly non-linear relationship in the input data. Historically, the most used function has been the sigmoid function, whose mathematical expression is given by:

$$\sigma(z) = \frac{1}{1+e^{-z}}.$$

The sigmoid function has the advantage of being limited between 0 and 1, which ensures that any neuron output in the network cannot reach extremely high values, thus making the training process more stable; it is also particularly useful for output neurons as it easily correlates to the probability of classification. However, for deep networks the so called ReLu (Rectified Linear Unit) function is most often preferred for a number of reasons: it is computationally simple while maintaining a sufficient grade of non-linearity and has proved to perform generally better.

$$ReLu(z) = \begin{cases} 0, & z < 0 \\ z, & z \geq 0 \end{cases}$$

A slightly modified version of this activation is the leaky ReLu, which introduces a low positive slope for negative values of input z; the derivative of the function is hence never zero, which can be beneficial for the learning algorithms that will be discussed below.

$$LeakyReLu(z) = \begin{cases} S * z, & z < 0 \\ z, & z \geq 0 \end{cases}$$

where s is a small positive slope, usually set to 0.01.

1.3.1.2 Learning process and optimization methods

The idea behind the learning process of an ANN is to iteratively feed labelled data as input, calculate the resulting output (ŷ) and its deviation (quantified through a loss function L) from the desired output (y) and adjust the network weights and biases in a way that minimizes this deviation. A generic cost function used for training is usually written as:

$$J(W, b) = \frac{1}{M} \sum_{i=1}^{M} L(y, \hat{y})$$

where M is the number of training example inputted to the network in a single pass. This is known as batch size and the single forward/backward pass is a training iteration; the presentation of the entire dataset constitutes a training epoch. Among many others, the most commonly employed cost functions are cross entropy (CE) and mean squared error (MSE) for classification and regression problems respectively.

$$J_{CE} = -\log(\hat{y})\, y \qquad J_{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y - \hat{y})^2$$

The algorithm at the core of the training process is known as backpropagation and, in its original formulation, is based on the gradient descent optimization method [20]. After each forward pass in the network, the loss function is evaluated and its gradient is propagated back through the network, updating the parameters by the following rules:

$$w := w - \epsilon \frac{\partial J}{\partial w} \qquad b := b - \epsilon \frac{\partial J}{\partial b}$$

Parameter ϵ, known as the learning rate, is used to control the entity of the parameters adjustment at each backpropagation: a higher value results in faster training, but may bring up problems as far as the convergence of the method is concerned, whereas a value too low could leave the training process stuck into a shallow local minimum. When the batch size is set to one (loss and backpropagation are computed after each example), this method is known as Stochastic Gradient Descent (SGD), while the case where batch size equals the size of the whole training dataset (loss and backpropagation are computed after one full epoch) is referred as Batch Gradient Descent: in practice an intermediate number is generally chosen, as a trade-off between the computing efficiency of the batched version that leverages vectorization and the frequent updates of the loss value of SGD. This last case is also frequently named Mini-batch Gradient Descent. A number of different optimization algorithms have been proposed as alternatives to GD [21], trying to achieve faster and/or more precise convergence, including the following:

- Gradient descent with momentum: it attempts to make the descent faster by factoring the gradient values from previous steps in the parameters update criteria during backpropagation, as shown below:

$$\begin{cases} w := w - \epsilon v_{dw} \\ b := b - \epsilon v_{db} \end{cases}, \quad \text{where} \quad \begin{cases} v_{dw} = \beta v_{dw} + (1-\beta)\frac{\partial J}{\partial w} \\ v_{db} = \beta v_{db} + (1-\beta)\frac{\partial J}{\partial b} \end{cases}$$

This is basically equivalent to using an exponential moving average of the latest gradients (which assigns an exponentially decaying weight to each averaged element, starting from the most recent) instead of the current value like the SGD did. This way, the optimization will be less prone to move in directions with oscillating gradients. The β hyperparameter controls how wide the moving average window actually is, and with the commonly used value of 0.9, roughly the most recent ten records in the average have a non-negligible effect.

- RMSprop: the concept is similar to that introduced by momentum, but instead of simply accounting for the average gradient, it introduces a damping factor, reducing the effect of oscillating gradients on the update rule. It is formalised as:

$$\begin{cases} w := w - \epsilon \frac{\partial J}{\partial w}\frac{1}{\sqrt{S_{dw}}} \\ b := b - \epsilon \frac{\partial J}{\partial b}\frac{1}{\sqrt{S_{db}}} \end{cases}, \quad \text{where} \quad \begin{cases} S_{dw} = \beta S_{dw} + (1-\beta)(\frac{\partial J}{\partial w})^2 \\ S_{db} = \beta S_{db} + (1-\beta)(\frac{\partial J}{\partial b})^2 \end{cases}$$

- Adam: its name stands for Adaptive moment estimation and it combines the two EMA terms of the previously described methods into one single update rule. However, instead of directly using the averages as computed above, it also applies bias correction that accounts for the effect of missing values in the first few iterations of computation. The final form of the update criterion is hence:

$$\begin{cases} w := w - \epsilon \dfrac{v_{dw}^{corr}}{\sqrt{S_{dw}^{corr}}} \\ w := w - \epsilon \dfrac{v_{db}^{corr}}{\sqrt{S_{db}^{corr}}} \end{cases}$$

  Here, the generic bias-corrected EMA is calculated as:

$$avg^{corr} = \frac{avg}{1 - \beta^t}$$

  where t is the current iteration number. As expected, the bias correction has a diminishing effect as t increases.

Considering all the topics above, the basic elements making up any ANN can be summed up in:

1. Neurons and their associated activation function.
2. Learning algorithm and optimization method.
3. Architecture, which defines the network size (number of layers, neurons per layer) and neurons interconnectivity.

1.3.2 Application to Image Processing

In principle, the most basic architecture described above, usually known as feed-forward architecture, where each neuron is connected to every other unit of the previous and following layers, could be applied directly in computer vision tasks by using the image pixels as features for the input layer. However, such an approach easily leads to an unsustainable increase in the number of learnable parameters (weights and biases) and it is hence viable only for low-resolution images. For example, a simple 512x512 RGB picture contains 786432 input features: considering a single hidden layer with 1000 neurons, the total number of parameters for this simple network already approaches one billion. As a consequence, a network like this is prone to overfitting without a consistent amount of training data, while still being too shallow to learn the complex patterns that are often part of imagery data (this brings to high bias, high variance models). The architecture that managed to solve these problems and first enabled to apply deep learning to computer vision with great success is the Convolutional Neural Network (CNN).

1.3.3 2D Convolution

As suggested by its name, the basic concept behind this architecture is the 2D convolution, an operator commonly used in many traditional image processing algorithms and whose mathematical formulation is usually written as:

$$y[i,j] = h[m,n] * x[i,j] = \sum_{m=-M}^{M} \sum_{n=-N}^{N} h[m,n]x[i-m, j-n]$$

where x is the input image, h is a [2M+1; 2N+1] matrix called "kernel" and y is the output image after convolution. Intuitively, convolution can be seen as the sliding of the kernel over the whole image, during which the value of the output image at the current position is calculated as the sum of the input pixels, weighted with the corresponding kernel values. There are two main approaches for handling the calculation when the kernel is centred near the border of the input image:

1. Compute the convolution only for the positions where the kernel is completely contained inside the input image. This is also called "valid" convolution. For an input size of [w; h] and a kernel size of [m; n], it generates an output of lower size [w-m+1; h-n+1].
2. Compute the convolution for all the input pixels, by adding p pixels, usually set to the value of 0, along the border of the input image (zero padding). When applying convolution on one image of size [w; h], padded with p pixels, and a kernel of size [m; n], the output size is [w+2p'-m+1; h+2p''-n+1]. The number of padding pixels is usually chosen so that the convolution operation leaves the image size untouched, which means:

$$\begin{cases} w = w + 2p' - m + 1 \\ h = h + 2p'' - n + 1 \end{cases} \longrightarrow \begin{cases} p' = \dfrac{m-1}{2} \\ p'' = \dfrac{n-1}{2} \end{cases}$$

   When these conditions are met, this is also called "same" convolution.

For deep learning applications, another very common parameter that controls the convolution behaviour is the stride, which sets the step for the kernel sliding over the image. Factoring in the stride s, the general output size becomes $[\frac{w+2p'-m}{s} + 1; \frac{h+2p''-n}{s} + 1]$.

1.3.4 Basic architecture of a CNN

The main difference between traditional convolutional filters used in image processing and those used in CNNs is that the kernel coefficients are not set beforehand, but they are part of the learnable parameters during training. A general CNN architecture is usually composed of two main macro blocks [22]. The first one takes an image as input and sequentially processes it into higher-level, lower-sized feature maps. For this reason, this part of the network can be seen as a feature extractor. The most common layers found in this block are:

- Convolutional (conv) layer: the core layer of any CNN, it applies convolution as described above between its input and a certain number of kernels. Its operation depends on four tuneable hyperparameters: kernel size, padding, stride and number of filters to apply. The output of a convolutional layer is a collection of feature maps, whose size in the first two dimensions depends on the combination of kernel size, padding and stride as formulated above, and a size in the third dimension corresponding to the number of filters of the layer. With a kernel size of [n; m] and a number of filters f, the number of learnable parameters is given by nxm weights and a bias term for each filter, for a total of $(n \cdot m + 1) \cdot f$ parameters,

irrespective of input volume size. The advantage of using convolutional layers in place of fully-connected layers of parameters needed is thus clear. Usually, the output volume of any conv layer is fed into a common activation function, typically ReLu.

- Batch normalization layer [23]: although not born in the context of convolutional networks and not being a standard block ubiquitously found in any CNN, this layer is an important component of many architectures. Its job is to normalize the batch of data outputted by a given layer; however, instead of computing a hard-coded normalization, this layer tries to learn the most suitable normalization directly from data. To do this, it brings two learnable parameters, γ and β, which are used to calculate the transform the input features as follows:

  $x' = \gamma \cdot x_{norm} + \beta$, where x_{norm} is the normalized input batch

  $$x_{norm} = \frac{x-\mu}{\sqrt{\sigma^2+\epsilon}} \mu = \frac{1}{M}\sum_{i=1}^{M} x_i \; \sigma^2 = \frac{1}{M}\sum_{i=1}^{M}(x_i - \mu).$$

  Basically, this layer first transforms the input batch into a zero mean and unitary standard deviation distribution, then it adds a new mean and variance that is adjusted as the training process advances. In addition, it also store the value of the moving averages of these two parameters through the iterations, making up two additional non-learnable parameters. Their value is used to apply batch normalization at inference time, when each input example is processed separately. When present, batch normalization can be placed both before and after the activation function of a convolutional layer; while there is no clear advantage of an approach over the other, this layer is most often placed before applying the non-linearity, as suggested by its introduction paper.
- Pooling layer: the main purpose of this layer is to reduce the spatial size of the input feature volume, to lower the parameters and computational effort needed for the deeper layers. This layer operates independently on every slice of the input volume and its behaviour is controlled by two hyperparameters: pooling region size (f) and stride (s). For an input volume of size [w'; h'; d'], a pooling layer outputs a volume of size

  $$\begin{cases} w'' = \dfrac{w' - f}{s} + 1 \\ h'' = \dfrac{h' - f}{s} + 1 \\ d'' = d' \end{cases}$$

  These two hyperparameters are usually chosen equal with a value of 2, which yields 2x2 non-overlapping pooling regions, with the effect of downsampling by a factor of two. For each pooling region, the output value is chosen by using one of the following pooling methods:
  1. Max pooling: the output is set to the maximum value in the pooling region. This is by far the most used alternative.
  2. Average pooling: the output is set to the average of the pooling region values. Historically, this was the first pooling method used, but nowadays it is almost always replaced with max pooling.
  3. L2-norm pooling: the output is set to the computed L2 norm of the values inside the pooling region.

Pooling layers do not introduce any learnable parameter in the network. The most recent CNN architectures have seen a reduced use of pooling layers in favour of convolutions with higher stride values.

The second block of a CNN takes the feature maps computed by the first block, flattens them to a feature vector and feeds it to a fully connected section for classification. The main layer types of this second block are:

- Fully-connected (FC) layer: the main component of non-convolutional feed-forward neural networks, where each neuron is connected to all units of the following layer. The only hyperparameter of this layer is the number of neurons. The total learnable parameters for one of this layer are: $(n^i + 1) \cdot n^{i+1}$, where n^i is the number of neurons in the ith FC layer and n^{i+1} the one of the following FC layer. The number of units of the first layer must be equal to the length of the flattened feature vector coming from the feature extracting block, while last layer units depend on the number of classes for the classification problem considered.
- Softmax layer: transform the last FC layer output vector $x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$ into a vector of probabilities $p = \begin{pmatrix} p_1 \\ \vdots \\ p_n \end{pmatrix}$, where each element $p_i = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$ is the probability of classification in the corresponding class.
- Classification layer: this layer defines the classification result as the class with the highest classification probability.

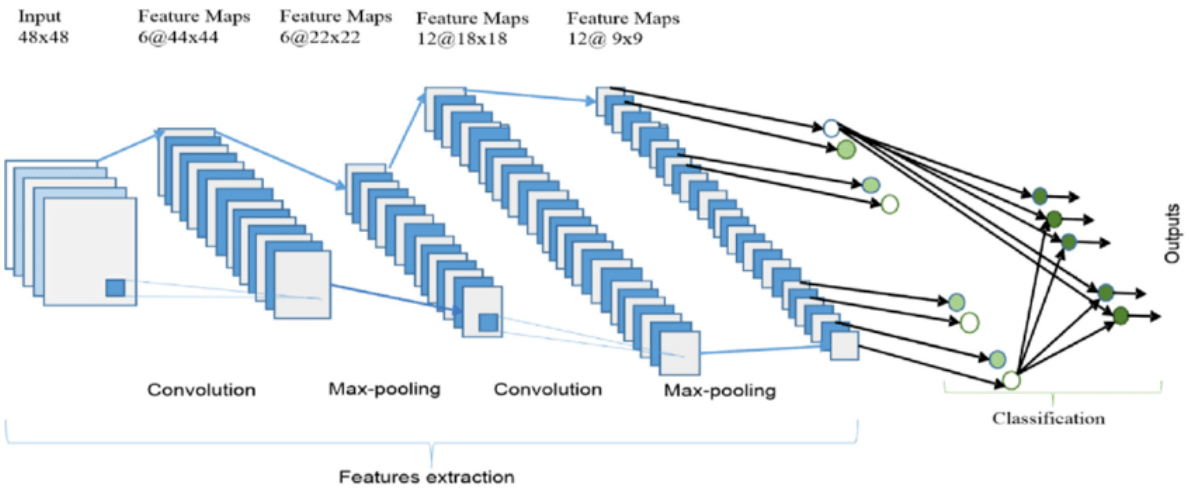


Figure 1.7: Typical CNN architecture

Materials and methods

2.1 Dataset

The image data used in this thesis work consists in a dataset proposed by Kumar et al. [24] as a standard to develop and validate nuclear segmentation algorithms on histological images. The dataset is composed by thirty 1000x1000 images, carefully extracted from TGCA (The Cancer Genomic Atlas) WSIs to contain a representative selection of different organs and structures. The original WSIs are H&E stained slides, captured at 40x magnification factor. To enhance the variability of nuclear appearance, the following criteria were used to extract any of the 30 images:

- Each image comes from a different patient biopsy, to include inter-subject variability.
- The biopsies are extracted from one of these seven organs: breast, liver, kidney, prostate, bladder, colon and stomach. This ensures the presence of different phenotypic traits.
- The biopsies used are prepared in 18 different hospitals, to account for variation due to different staining protocols.

The ground-truth is represented by the separate segmentation of each nucleus, for a total of 21623 objects, and the manual annotations quality has been reviewed by an expert pathologist; the authors report the number of incorrect annotations for any given image to be lower than 1% of the total. Following the authors suggestion, the dataset was split as follows:

- Training set (13 images, from four of the seven organs): used for the training of both the segmentation and detection networks.
- Validation set (3 images, same organs of the train set): used to compute the training metrics at each epoch of the training process.
- Test set 1 (8 images, same organs of the train set): used to assess the framework ability to generalize learned tasks to images never seen by network during training.
- Test set 2 (6 images, from three different organs): used to further benchmark generalization by using nuclei from organs that were excluded from training.

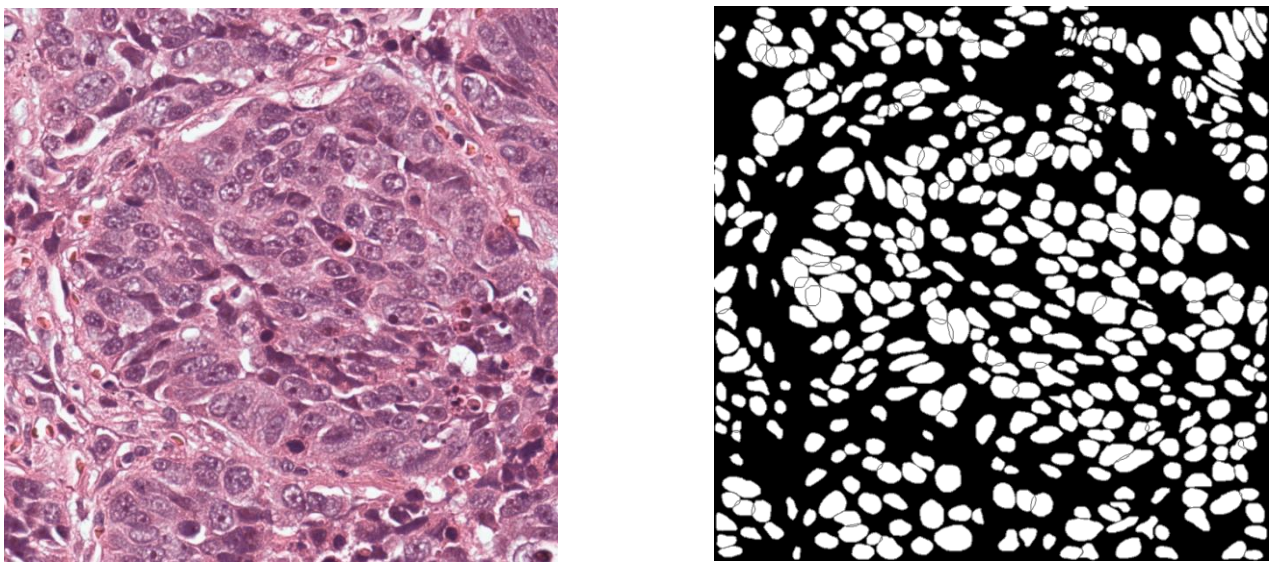


Figure 2.1: An example of 1000x1000 image from the dataset with its associated ground truth annotations

In addition to the dataset described above, the starting material for this work already includes the softmax probability maps generated by a U-Net like segmentation network. This network was trained on a 3 classes segmentation problem (nucleus, nucleus border and background) on the above mentioned training set, as already proposed by Kumar. In particular, the softmax is coded as a 3-channels RGB image, where for each pixel the red value is the probability of being part of the background, the green value is the probability of constituting a nuclear border and the blue one the likelihood of being part of a nuclear body. Being a graphical representation of probabilities, the sum over the three channels for any given pixel always amounts to one. The introduction of the border class should make it easier for the network to learn to separate close distinct nuclei from each other, however many instances where two or more nuclei segmentations are fused without any border are still observable.

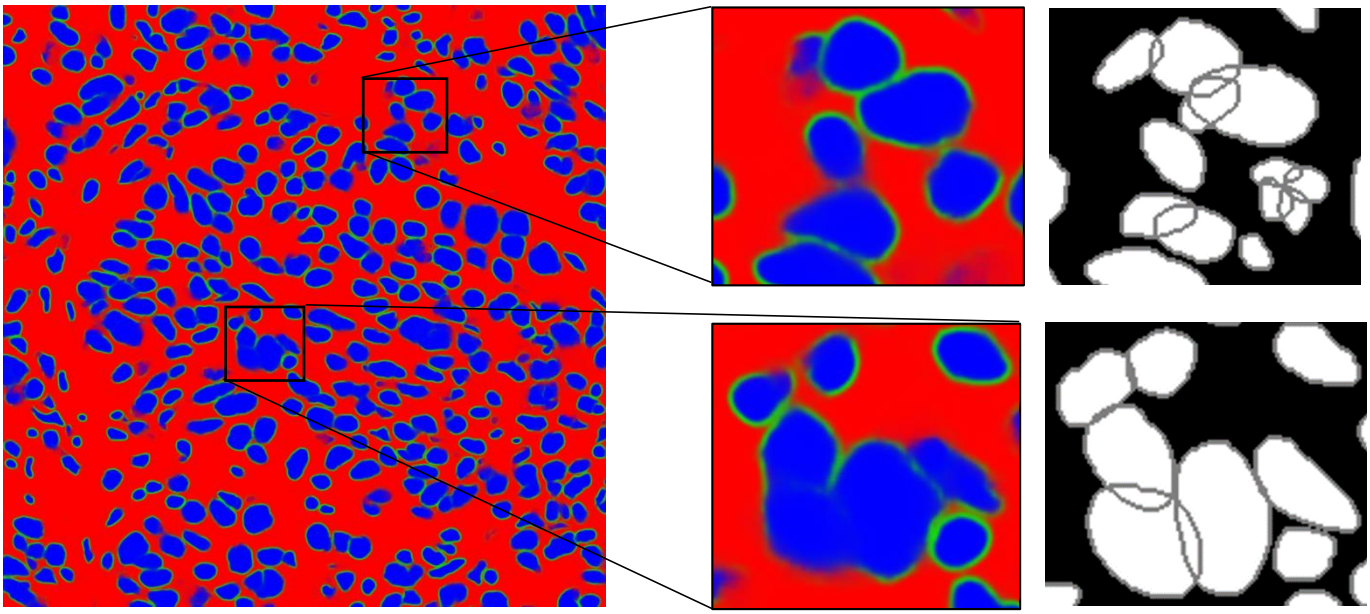


Figure 2.2: Fused nuclei in the softmax probability map

This dataset was the subject of the MoNuSeg 2018 Challenge [25], whose results are reported in [article]. In this case, though, all 30 images of this dataset were used as training data, while a separate test set of 14 images was provided to participants after the submissions deadline.

2.2 Previous work

To the best of my knowledge, no available published work tries to combine deep learning frameworks with traditional post-processing techniques in the same way explored in this thesis (evolution of active contours on softmax probability maps and their initialization through an object detection model). Outside of the medical imaging field, the integration of deep learning methods with active contours has been recently addressed by the work of Rupprecht et al., who developed an interactive segmentation method that they named “Deep Active Contours” [26]. Given an evolving geometrical active contour, a class-specific convolutional neural network is trained to predict, for each point of the contour, a vector pointing to the closest point on the boundary of the object to be segmented. The vector field originated from all the contour points is subsequently used to iteratively evolve the contour. The authors also demonstrate the potential applicability of their method to the medical field by successfully segmenting the left ventricular cavity of MRI

acquisitions. More recently, Marcos et al. developed and end-to-end trainable framework for the segmentation of buildings on aerial images [27]. This framework, named DSAC (Deep Structured Active Contours), aims to combine the typically high detection rate of CNNs with the geometric correctness of snakes. In contrast with the previously described paper and the approach used in this thesis (using deep learning initialized snakes as a post-processing step for the segmentation network), the authors managed to integrate the contour evolution into a loss term, thus making it a central part of the CNN training process.
In the histological imaging field, Kumar et al. presented a segmentation framework based on their own proposed dataset [24], the same used for this thesis. Their method is based on a CNN with three cascaded convolutional/pooling blocks and two FC layers, that classifies each pixel in one of three classes (inside/outside of a nucleus and nuclear border). To reach the nuclear segmentation starting from the three-class softmax map outputted by the network, the "inside probability" is thresholded to obtain nuclear seeds which are then expanded using region growing. The proposed idea of explicitly introducing a separate class for the nuclear border helps with the segmentation of crowded and touching nuclei.

2.3 Pre-processing

2.3.1 Color normalization

The images provided already went through a pre-processing step of color normalization, to minimize the color variations that purposely affect the dataset. This is a standard pre-processing step when dealing with histological image analysis, as the strong color variability could otherwise excessively bias the algorithm learning process. Color normalization includes a wide set of techniques, all aimed at compensating the color variations on a given group of input images to make them as coherent as possible with a reference image. The choice of the normalization technique to implement is quiet important, as it can significantly influence the performance of any classifier or detector later applied and it should be evaluated considering a number of factors: impact on the overall image quality and details, degree of coherence with the original coloration, presence of artefacts, computation complexity and others. Due to the importance of this step on the whole histopathological image analysis process, a great number of techniques have been proposed in the recent years [28]. Although now mainly outdated, some of the most historically important among the others are:

- Histogram Specification [29]: one of the first attempt to color normalization on histological images. It can be rather limited and yield unsatisfactory results when the statistics of source and target images are highly dissimilar.
- Color Deconvolution: known as the first methods to successfully separate stains in a histology image. This can be particularly interesting because each stain usually highlights particular biological structures.
- Spectral Decomposition [30]: employs Non-negative matrix factorization to decompose the input image, but it presents scaling ambiguities, as the solution to NMF is not closed.
- Macenko [31]: it is a fast and automatic method to perform stain separation, but a consistent amount of information from source image can be lost, which can be unacceptable for certain analyses.

In more recent times, some deep learning related techniques have also been proposed [28] and evaluated; the most promising approaches are based on Generative Adversarial Networks (GANs) and Cycle-GANs.

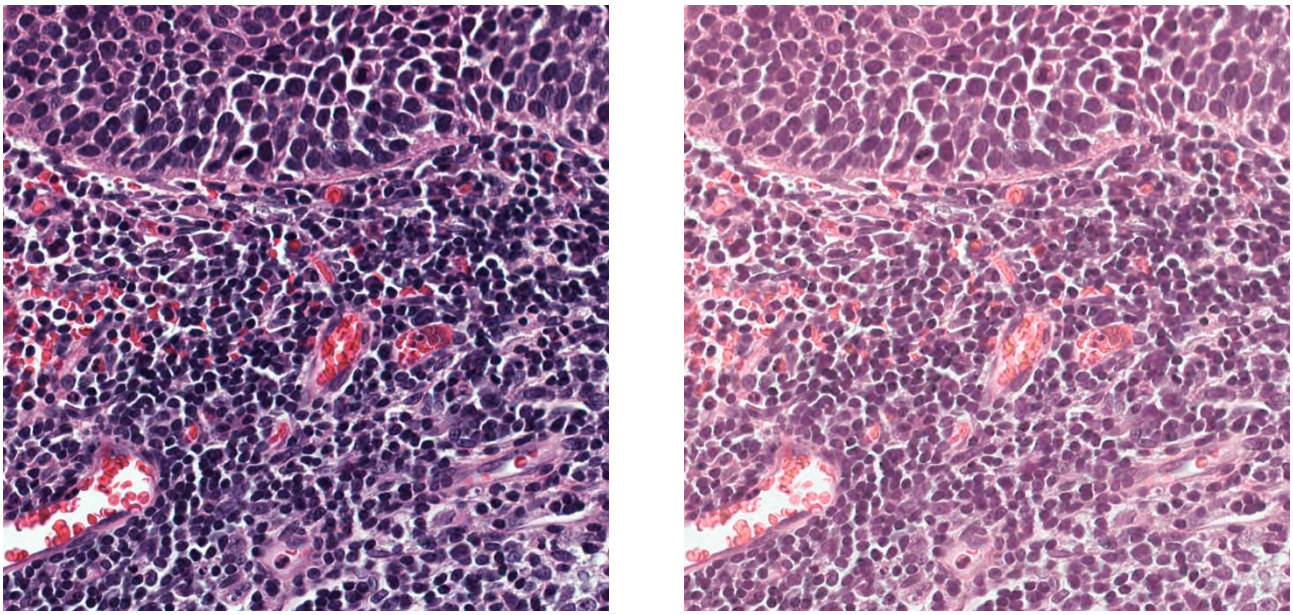


Figure 2.3: An image from the second test set before and after color normalization

2.3.2 Patch extraction

First of all, bounding box information is extracted from the annotation data structure associated with the dataset. Each bounding box is expressed with its centre coordinates, its width and height, all normalized between 0 and 1 in respect of the total image (or patch) length. Afterwards, before proceeding with the training of an object detection model, the dataset is offline preprocessed to be divided into square patches through a dedicate Matlab script. Each patch is saved as a separate image, along with the coordinates of any bounding boxes on the whole image whose centroid lies in the patch itself and the information needed to correctly reconstruct the original image during subsequent model inferencing on the patches (coordinates of the higher left pixel and size of the patch). The parameters that can be set to control the way that the square patches are extracted are:

1. Patch size (N): when N is selected to any value below half the size of the image (500), the maximum number of non-overlapping patches of size NxN is extracted to cover up the full image. On the other hand, when the selected size exceeds this threshold, four patches are always extracted, one for each corner of the input image.
2. Extraction at grid intersections: optionally, the patches centred on the grid intersections formed by the borders of the previously included patches can also be extracted. The idea behind this possibility is to increase the number of training examples, providing a first offline form of data augmentation.

Two representative examples of the described patching process are reported below, with a selected patch size of 250x250 and 640x640 respectively.

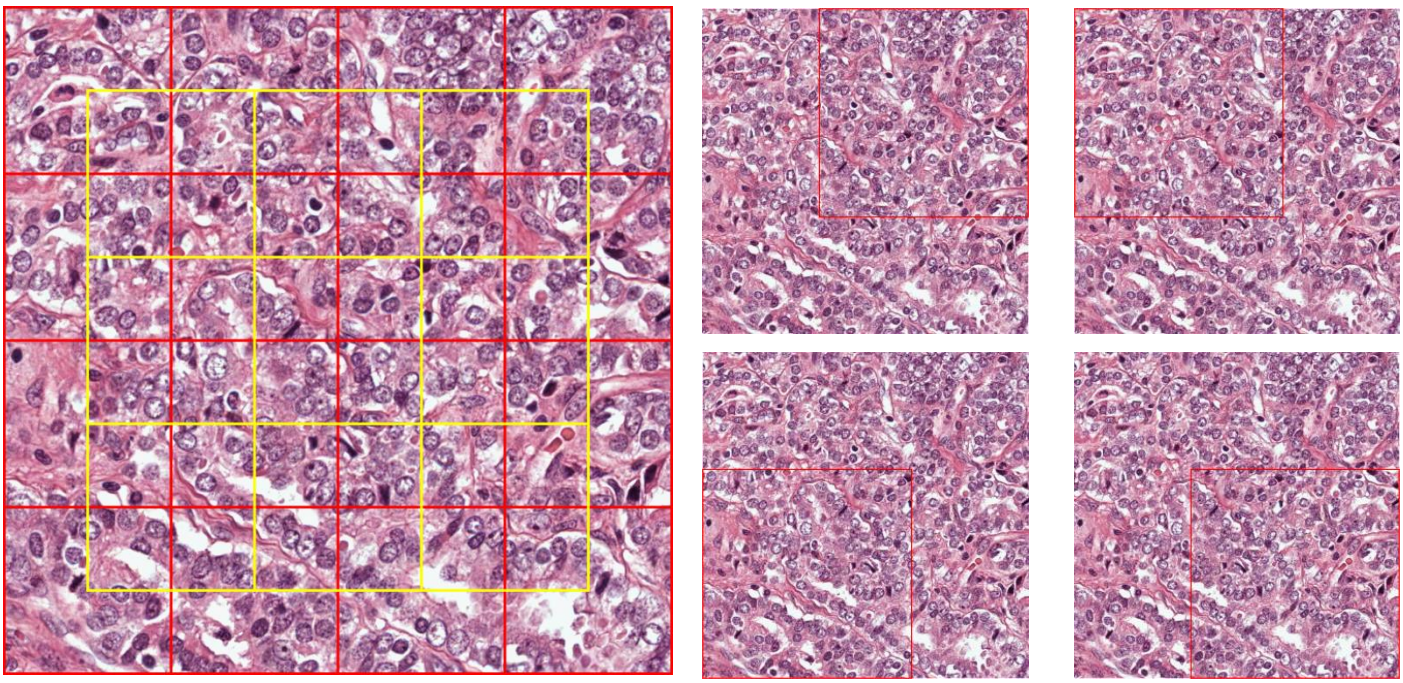


Figure 2.4: Patch extraction from the full image (on the left 250x250, on the right 1000x1000

2.4 Object Detection

The first step of the proposed method is the training of an object detection model, through the same 13 training images used for the U-Net training. When the relevant architectures linked to this work make use of more advanced layers that those discussed in the introduction chapter, those will be briefly explained in dedicated subparagraphs.

2.4.1 Computer Vision tasks

Object detection is a computer vision task that has seen a very strong growth in the recent past. Considering the complexities associated with image data and the breadth of the field, computer vision is rich of challenges that are now approachable through the use of deep learning [32]. Some of the most common and relevant for this work are:

- Image classification: it consists in assigning a single label to the whole image, based on its content. This is one of the first addressed and relatively simple computer vision tasks and it can be attained with the basic CNN structure described in the introduction.
- Classification and localization: this is similar to the previous task, but instead of just recognizing the main subject of the image and assigning a proper class, it also highlights the spatial location of the object using a bounding box. This is still limited to a single object per image.
- Semantic segmentation: the whole input image is segmented by assigning a class to each pixel. No notion of object is introduced, so this approach may not be suitable for the segmentation of crowded or overlapping subjects, when distinguishing between instances is

needed. This task can be successfully performed with deep learning segmentation architectures such as U-Net.

- Object detection: the goal of object detection is to detect every instance of different objects of interest in a picture, as well as to highlight its spatial localisation with a bounding box. Considering the need to identify an unknown number of instances of possibly very different object classes, object detection is an inherently harder task compared to those listed above.
- Instance segmentation: just like object detection, all instances of different objects are detected, however, in addition to that, each instance is also segmented at pixel level.

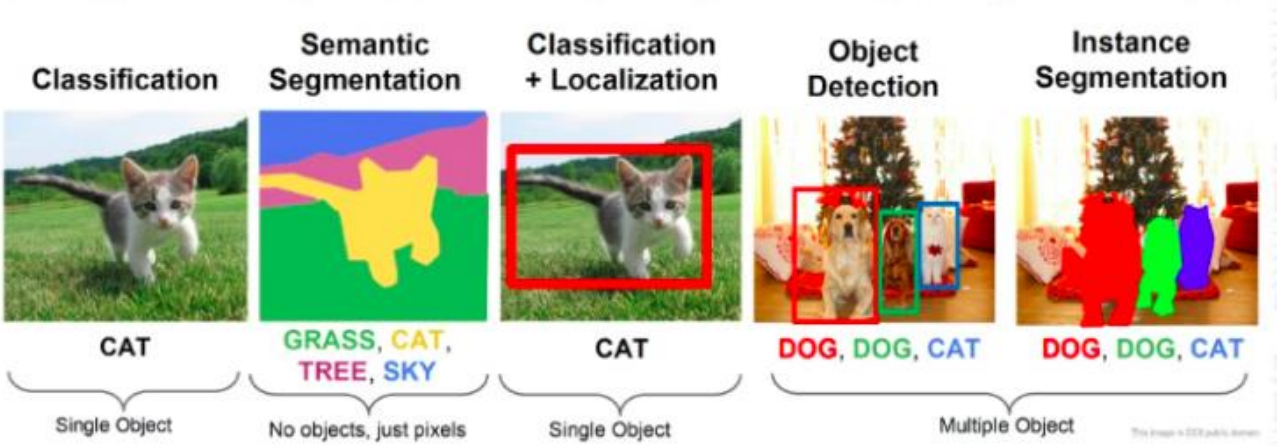


Figure 2.5: Examples of different computer vision tasks

2.4.2 One-stage and two-stage object detectors

Object detection models typically fall in two main categories, in respect of their approach to the detection problem: one-stage and two-stage detectors [33]. One-stage detectors directly treat detection as a regression problem: starting from the input image, they output a collection of bounding boxes that can be positioned on any part of the image, in terms of their coordinates and object class. During backpropagtion, the coordinates and classes computed during the forward pass are used along with ground-truth annotations to compute some kind of paired regression and classification losses. One-stage detectors have the distinctive advantage of being very fast at evaluation time, making real-time inference possible even on consumer grade hardware. The most representative models of this object detectors type are YOLO (acronym for You Only Look Once), SSD (Single Shot Detector) and RetinaNet. Two-stage detectors on the other hand split up the detection process in two separate phases:

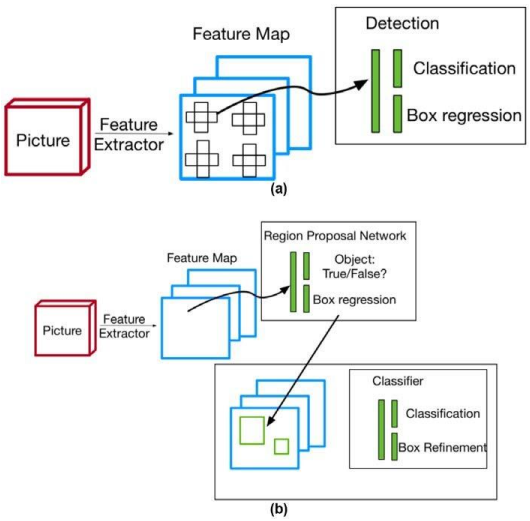


Figure 2.6: One-stage and two-stage object detectors

1. Proposals generation: during this phase, the model identifies regions on the input image that are most likely to contain an object. These proposal regions (or Regions of Interest; ROI) can either be performed by specific algorithms (such as selective search) for former detectors or by a neural network itself (usually called Region Proposal Network; RPN) in more recent

models. Especially when using dedicated algorithms, this stage is usually the weak part of two stage-detectors when it comes to computational time.

2. Classification: the proposals from the previous step are classified into object classes or background by a dedicated network. The network may include a regression branch to further refine the bounding box position in the proposed region.

Due to their indirect nature, two-stage detectors are fairly slower than one-stage solutions, however they can reach better detection performance on average. Some import two-stage detectors are R-CNN and its evolutions (Fast and Faster R-CNN) and FPN (Feature Pyramid Network). The above is just a general principle of working, which may of course vary significantly even between two detectors of the same family.

2.4.3 Common datasets and evaluation metrics

Just like any other computer vision task, in particular when deep learning methods are involved, availability of abundant and high quality annotated data is crucial to obtain good performances and, even more importantly, to fuel and accelerate the development of new relevant techniques and algorithms in the field. Indeed, even the first detectors with concrete applications only dates back to the first years of the new century and the time span between new milestones still laid in the order of several years. These first successful algorithms, such as the Viola James detector and Deformable Part Models (DPM), are based on the extraction of statistical handpicked features and on traditional image processing techniques based on sliding window approaches. It was only in the last decade, with the advent of CNNs and the public availability of large and well balanced general-purpose datasets, that deep learning could see its first significant adoption in the object detection field, with an incredible reduction in the time between new detectors setting the bar higher for state-of-the-art detection. Many datasets have since become a standard benchmark to assess performances of newly proposed methods in a reproducible way and are easily found in any object detection paper. In the following list, the most common will be cited, along with their salient peculiarities [33][34]:

- PASCAL Visual Object Classes (VOC): subject of yearly challenges between 2005 and 2012, it was one of the earliest large bounding box annotated datasets in the computer vision community. In particular, two versions have become a standard benchmark in object detection: VOC 2007 and VOC 2012. They feature 5k/11k training images and 12k/27k annotated instances respectively, belonging to 20 classes. With the release of new and more challenging datasets, VOC is now mainly used to provide a fair comparison with previous related works.
- ILSVRC: based on the well-known ImageNet dataset for image classification, it provides a higher variety than VOC, thanks to its 200 classes, spread through 517k images and 534k annotations.
- MS-COCO: probably the most challenging general-purpose dataset available as of today for detection tasks, it offers 80 object categories, which is less than ILSVRC, but with a fairly higher instances to images ratio (897k annotations over 164k images). Moreover, it contains more small objects (with an area less than 1% of the whole image), which usually represent a higher detection challenge. Each annotation also include the complete instance segmentation of the object, making it ideal for models like Faster RCNN that can also perform

this kind of task. Thanks to its peculiarities that make it a good sample of real world images, COCO has become the current standard of model benchmarking.

- Open Images: released in 2018, it pushes the amount of data in a single dataset to new highs. In fact, it is composed by almost 2M images, with a number of annotations just below 16M and 600 separate object classes.

Of course, many other field-specific datasets exist and are insightful to compare proposed works in that relevant field, just as the dataset used in this thesis is one of the standards for nuclear segmentation. After introducing the datasets, another important point that needs to be discussed to achieve truly reproducible comparisons is the evaluation metrics employed to evaluate performance. In recent times, the most used metric for object detection has undoubtedly been the mean Average Precision (mAP) [33][34]. It is based on Average Precision, which is defined as the average value of precision at different levels of recall. Hence, it is computed by estimating the AUC (area under curve) for the precision-recall curve. All the AP values calculated separately on each class are then averaged to obtain the final mAP value.

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i \quad ; \quad AP_i = \int_0^1 p_i(r)dr$$

Where p and r indicates precision and recall respectively, computed with their typical formulations:

$$p = \frac{TP}{TP + FP} \quad ; \quad r = \frac{TP}{TP + FN}$$

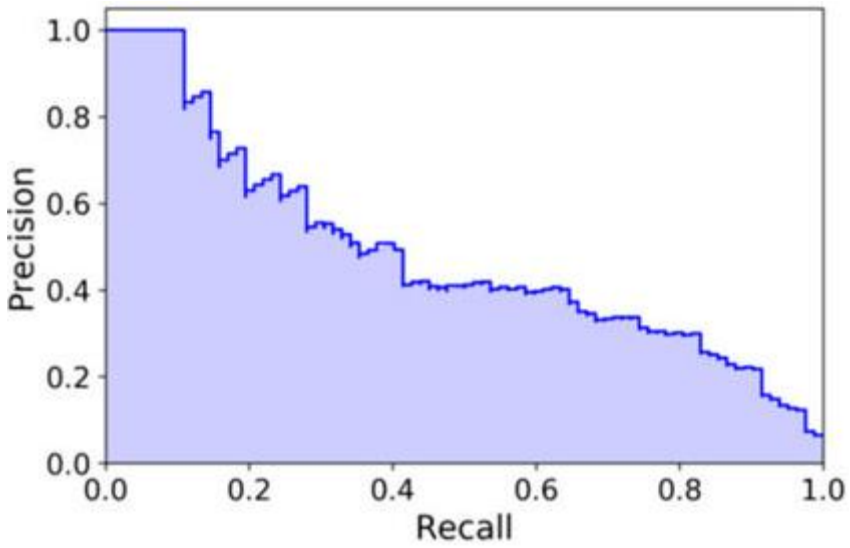


Figure 2.7: Precision-Recall curve - the blue area yields the AP value

However, an important factor to consider when performing mAP computation is how a certain prediction is considered as a TP or a FP. To do this, Intersection over Union (IoU) is employed: it is defined as the ratio of the area of intersection between a given predicted box and the ground-truth box and the total union area. Typically, an IoU of 0.5 is considered the minimum threshold for an accettable localization, while anything under that is considered a miss. In fact, this standard value has been used to compute mAP since the introduction of VOC challenge and the metric is also indicated as mAP@0.5. In the last few years though, with the introduction of COCO dataset and

justified by the increasing localization capability of detectors, the practice of considering detection quality at stricter IoU values has been increasingly common and the standard is now to provide a mAP calculation averaged over multiple IoU threshold values, most often from 0.5 (coarse localization) to 0.95 (almost perfect localization). This metric is usually reported as mAP@0.5.0.95 and, despite being at first criticized by some researchers, this is now the metric on which state-of-the-art performance is defined.

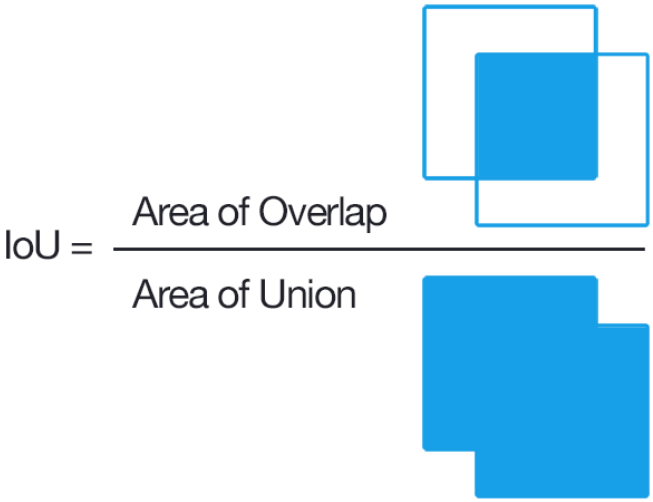


Figure 2.8: IoU definition

2.4.4 Non-Maximum Suppression

Non-Maximum Suppression (NMS) is a core algorithm in computer vision, especially for object detection tasks. Traditionally, it is used as a post-processing step after model inference, but in some cases it is also applied during training (like in Faster RCNN, where it is applied after RPN to reduce the number of proposals) and recently some papers pushed further by making it a part of the end-to-end learning process of the model. Most algorithms and models for detection, especially those that rely on region proposals (such as two-stage detectors), generate a great number of predictions even when the number of objects on an image is limited. Each prediction is accompanied by a score, indicating the confidence of the model that the proposed bounding box corresponds to an actual object of a particular class in that position. Consequently, a model outputs multiple bounding boxes in close proximity when trying to predict the localization of a single object. Of course this can be suboptimal, especially for applications where a good estimation of the number of instances is important, and it strongly affects the general precision offered by that model. NMS tries to address this problem and reduce all the duplicate predictions to a single, most representative box. In its basic form, also referred as “Greedy NMS”, it is an iterative algorithm that takes the predicted bounding boxes and their associated confidence scores in input and returns a new, reduced set of predictions. As a first step, predictions are sorted in descending score order; in a single iteration of the algorithm, the first prediction (highest-scoring, also called maximal prediction) is selected and moved to the final predictions set. The IoU of this prediction with all the others is then calculated and any box with an IOU greater than a certain threshold with the selected box is discarded. The process then repeats by picking a new maximal box from the remaining list and it only stops when all the original boxes have been either selected or removed. The computational complexity of NMS

for a set of N boxes is $O(N^2)$, so all the predictions with a confidence below a minimum value are usually removed prior to the actual application of the algorithm. The reasoning behind NMS is that for each object in the input image, the detector is expected to output a cluster of boxes with different confidence in its surroundings: NMS considers any prediction with enough overlap with the most confident one to just be a weaker detection of the same object and suppresses it. When this is the case, NMS can lead to significant precision increments without any impact on the recall. However, in a general scenario where separate objects can be close and have a partially overlapping bounding boxes, the choice of the IoU threshold is not trivial: a threshold too high can defeat the purpose of the algorithm, leaving a significant number of false positives, while a low value starts to cause the suppression of close true positive detections due to their partial overlap. Therefore, the possible trade-off between increased precision and lowered recall must always be accounted when setting the threshold value. Moreover, the highest confidence score box is not guaranteed to be the most accurate prediction for a given ground-truth object [33].

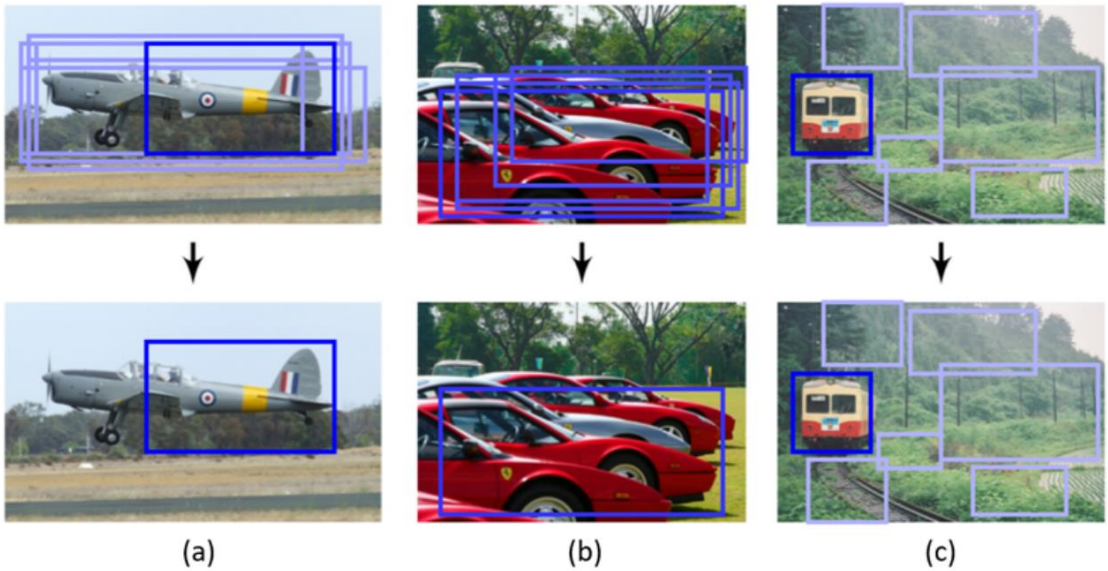


Figure 2.9: Typical NMS shortcomings: a) The highest confident box is not the most fitting. b) Suppression of close TPs. c) Non overlapped FPs.

Despite Greedy NMS still being the most generally used post-processing method, many variations have been developed through the years, trying to overcome the basic implementation limitations; these can roughly be grouped in three categories:

- Modifications to the greedy approach: these methods apply the same iterative process of tradition NMS, but try to introduce more solid suppression criteria than the simple IoU evaluation. A fitting example is represented by an approach called "Soft-NMS" [35]: when a prediction overlaps with the maximal box for a higher portion than the IoU threshold set, its confidence is reduced through a rescoring function, instead of being removed directly. The score decay depends on the entity of the overlap and is computed using a Gaussian penalty function: $s_i = s_i e^{-\frac{IoU(M,b_i)^2}{\sigma}}$, where M is the highest-scoring prediction, b_i and s_i are a generic box compared to M and its corresponding score. By doing so, the predictions with a consistent overlap are heavily penalized, while potential true positives close to each other

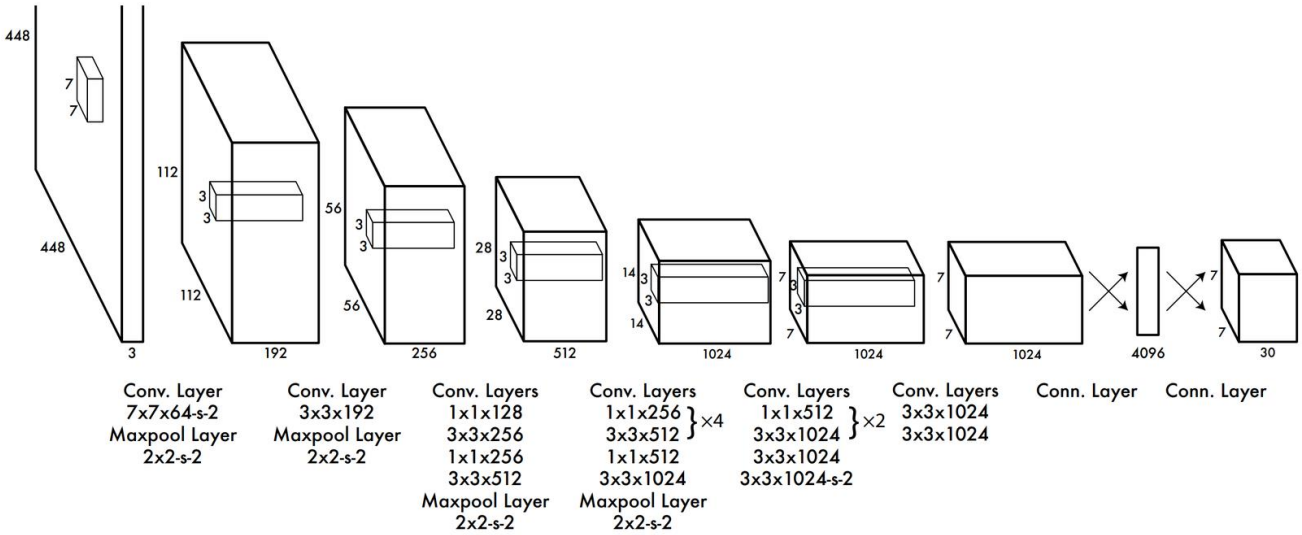
still have a chance of being maintained if the inference threshold used in model evaluation is low enough. Despite being a fairly simple variation over traditional NMS, improvements of about 1% AP@0.5:0.95 are reported on COCO and VOC 2007 datasets, with two-stage detectors obtaining greater benefits than single-stage solutions.

- Bounding boxes aggregation: instead of keeping only one highest-confident prediction and removing the others nearby, these methods try to use the information coming from each bounding box to combine them into one final detection. One of the most recent examples is represented by Weighted Boxes Fusion (WBS). Predictions given in input to this method are first clustered by using a simple IoU criterion and for each of them a representative box, whose confidence score is equal to the mean of the cluster's confidence scores and coordinates are computed as a score-weighted average of all the boxes that compose the cluster. WBS proved to be mostly useful when processing multiple predictions coming from model ensembling, whereas its application as a direct NMS replacement on a single model output is usually slightly detrimental [36].
- Integration of NMS into the learning process: considering its importance in almost all object detection pipelines, NMS is directly integrated in the training workflow of the model, in place of being used as a post-processing step after model inference. Some approaches follow the reasoning behind Soft-NMS of decaying confidence scores of close predictions, but instead of doing that in a predefined way, they introduce a specific convolutional network to perform this task [37]. In other works, such as in [38], one or more NMS related cost terms are formulated and added to the loss of a standard detection architecture. The advantage of these end-to end solutions is that they do not require any tune of parameters after their training, which is one of the main problems for traditional NMS.

Regardless of its great impact on detection results and of any further enhancement that might be introduced, NMS remains a final step in the detection pipeline, as it cannot help in the removal of false positive instances that are not due to double detections, but caused by the model itself identifying a portion of background as an object.

2.4.5 YOLO architecture

In the context of this thesis, a YOLO object detection model was chosen. Before describing the model training approach used and its results, a brief introduction to the algorithm is reported. YOLO is a single stage object detection model proposed by Redmon et al in 2016 [39] and it is the real first one-stage detector to reach real-time performance (at least 30 fps inference), without a big performance trade off when compared to contemporary state of the art two-stage detectors. The distinctive point of YOLO lies in reframing object detection as a single regression problem, which predicts bounding boxes and their associated class labels directly from the whole input image. This allows to approach the detection problem using a single standard CNN architecture, consisting of alternating convolutional and max-pooling layers followed by two FC layers as pictured below.



It employs 1x1 convolutional layers, which might at first seem like a meaningless operation, but it is actually common practice in many modern architectures. In fact, when applied to a 2D matrix it is nothing more than a multiplication for a constant, but when used on a volume as it is the case for feature maps, it actually computes an element wise multiplication, thus reducing the volume to a single 2D output. As any convolutional layer, an arbitrary number of 1x1 filters can be applied, so for an input of size [W; H; D], the output will be [W; H; D'], where D' is the number of 1x1 kernels applied by the layer. 1x1 convolutions can then be seen as a convenient way of shrinking or expanding the feature map depth-wise without changing the first two dimensions, just like strides on convolutional layers and max-pooling are used to control height and width of representations through different network levels. The concept of 1x1 convolutional layers was first introduced by Lin et al, which referred to it as a "Network in Network" approach [40], as each 1x1 kernel sliding on the input can be seen as a neuron of a FC layer connected to each 1x1xD portion of the volume.

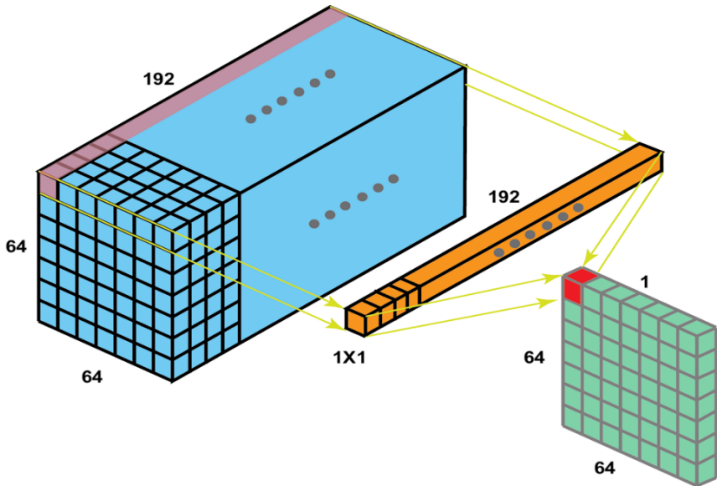


Figure 2.10 1x1 convolution

The CNN takes a 448x448 RGB image in input and outputs a volume of size 7x7x30 containing the bounding boxes predictions. The basic steps followed by YOLO to generate such predictions are:

- Divide the input image into a grid containing SxS cells. Each grid cell will be responsible for the detection of one object whose center lies inside the cell.

- For each cell, predict B bounding boxes and associated conficidence scores. That score factors the detector's confidence that the box actually contains an object as well as the confidence in the box localization accuracy; it is formulated as the product of the probability of containing an object with the intersection over union of the prediction with the corresponding ground-truth box: $conf = \Pr(Object) \cdot IoU_{pred}^{truth}$. Each bounding box is made up of five predictions (x, y, w, h, conf): the pair of coordinates (x, y) is the centre of the box, relative to the grid cell borders, w and h are box width and height, normalized by the whole image size, and $conf$ is the confidence score as above. This way all the box predictions for any given cell are normalized in the range [0; 1], which make the optimization of the regression loss easier.
- Prediction of C conditional class probabilities for each grid cell. These probabilities are conditioned on the cell containing an object and are hence expressed as $\Pr(Class_i|Object)$.
- During test time, the conditional class probabilities are multiplied with the individual confidence score of boxes inside the same grid cell to obtain the final class-specific confidence scores for each box.

$$conf_{final} = \Pr(Class_i|Object) \cdot \Pr(Object) \cdot IoU_{pred}^{truth} = \Pr(Class_i) * IoU_{pred}^{truth}$$

- All the predictions from the last FC layer are thus rearranged into an SxSx(5B+C) tensor: the first two dimensions come from the grid cell subdivision, while the third depends on the five predictions of each box (center coordinates, box size and confidence score) and the classification scores of the grid cells. In the original paper implementation, evaluated on VOC dataset, the input image is divided in 49 cells (S = 7), each predicting two bounding boxes (B = 2) to detect object of twenty separate classes (C = 20). As a consequence, the final tensor yielded by the described CNN architecture is 7x7x30.
- At the end of the inference phase, NMS is applied to remove possible double detections, due to big objects extending over more than one grid cell.

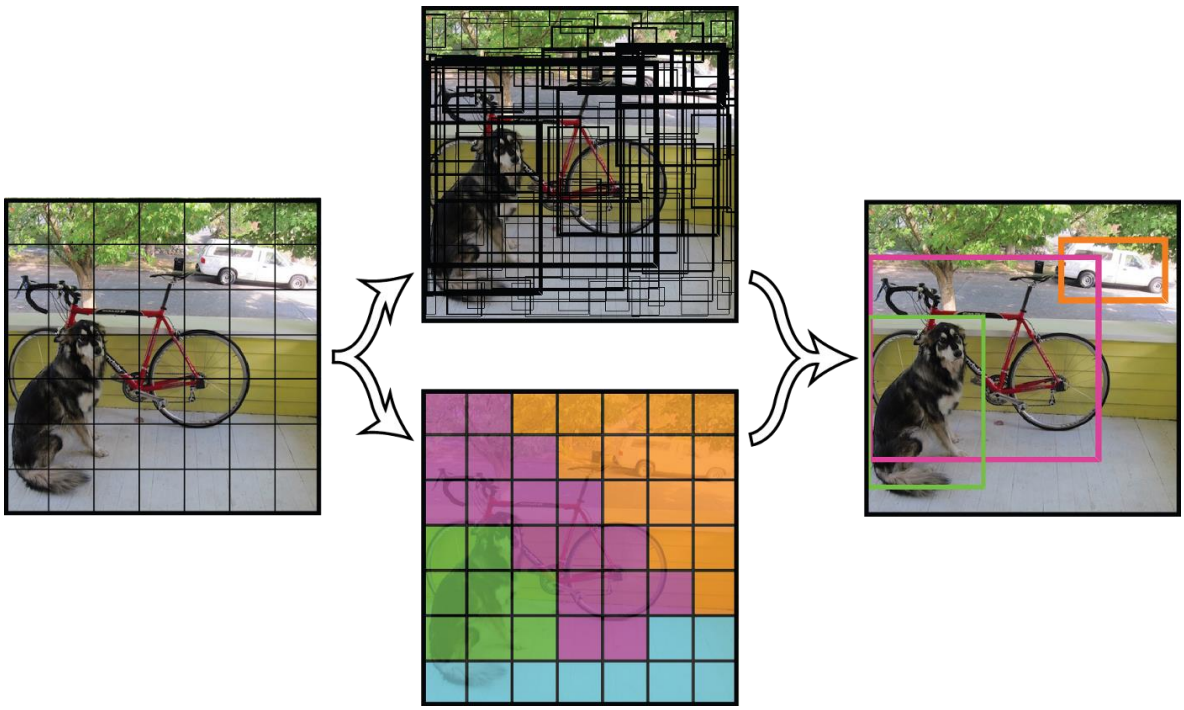


Figure 2.11: Working principle of YOLO detector

The loss function used by YOLO is a multi-part sum squared error function, which is naturally suited for regression problems and easily optimized. It consist of a localization and classification contribute, so that the total loss can be written as: $L = L_{loc} + L_{cls}$, with:

$$L_{loc} = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 + \left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2 \right]$$

$$L_{cls} = \sum_{i=0}^{S^2} \sum_{j=0}^{B} \left(1_{ij}^{obj} + \lambda_{noobj} 1_{ij}^{noobj}\right) \left(C_i - \hat{C}_i\right)^2 + \sum_{i=0}^{S^2} \sum_{c \in C} 1_i^{obj} \left(p_i(c) - \hat{p}_i(c)\right)^2$$

In the equations above:

- 1_{ij}^{obj}, 1_{ij}^{obj} and 1_{ij}^{obj} are indicator functions that can only assume 0 and 1 values: 1_{ij}^{obj} and its complement 1_{ij}^{noobj} indicate whether the j-th bounding box in i-th grid cell contains an object, 1_i^{obj} whether the i-th grid cell includes any object.
- The first term of the classification loss accounts for the model's ability to distinguish an actual object from the background, while the second one evaluates the objects classification into the correct classes.
- The multiplicative constants λ_{coord} and λ_{noobj}, set to 5 and 0.5 respectively, are used to increase the localization term impact on the total loss and to reduce the weight of boxes that does not contain any object on the first term of the classification loss, considering that the number of grid cells that does not contain any instance is fairly high for a generic image.
- All the terms inside summations except the indicator functions are not indexed over the j index (bounding box number inside a specific grid cell), because during training only the box with the highest IOU in each grid cell (I index) is used in the cost function computation.
- The square root of the box width and height is used in the localization term so that the penalization accounts for box size (small errors have a higher impact when the true box is small).

All the innovations introduced by YOLO enabled it to be both faster and much more performant than any previous window-based single stage detector, reaching more than 63% mAP at 45 FPS inference on VOC 2007 dataset, using a Titan X GPU. Compared to state of the art two-stage detectors like Faster RCNN, YOLO shows better generalization thanks to its ability to factor in the whole image context when making predictions, reducing the number of background false positives and making it a reasonable choice for tasks involving a high degree of variability, like that addressed in this thesis. Despite the numerous upsides, this original implementation still has different limitations, some of which make it unideal for the purpose of this work. To the high model precision corresponds a recall generally lower than two-stage models, along with a higher error associated with boxes spatial localization. The model can only locate two bounding boxes with a shared class per grid cell, so its performance is greatly reduced in case of small, grouped objects, which is the reason that lead to the introduction of an object detection model within the proposed framework in the first place.

2.4.6 Evolution of YOLO

Following the publication of the first YOLO paper and its great impact in the field of computer vision, many new research works were published in a relatively short span of time in relation to one-stage detection, be they brand-new takes on the detection problem, such as SSD and RetinaNet, or improvements on the first YOLO algorithm. The same author of the original implementation published two new versions, YOLOv2 and YOLOv3, in 2017 and 2018 respectively. Other two newer implementations, YOLOv4 and YOLOv5, were released during 2020 and, while not officially linked with the original work, they maintain the name and its philosophy. A description of the most significant differences between the successive versions of YOLO is now reported.

2.4.6.1 YOLOv2

YOLOv2 was proposed to address the main shortcomings of the original implementation, namely the low number of detectable instances per image (49), the high number of localization error on TP boxes and the lower recall compared to contemporary two-stage detectors, all this while maintaining the great computational simplicity at its core [41]. The be able to make YOLO even faster despite the modifications discussed below, the authors replaced the original architecture with a whole new custom backbone called Darknet-19. This architecture, whose structure is summarised in the table below, is again based on alternating convolutional (19 in total) and max-pooling layers. It only requires 5.5 billion floating operations per single image as compared to 8.5 billion for the previous YOLO network, still reaching better classification accuracy on ImageNet.

Type	Filters	Size/Stride
Convolutional	32	3×3
Maxpool		$2 \times 2/2$
Convolutional	64	3×3
Maxpool		$2 \times 2/2$
Convolutional	128	3×3
Convolutional	64	1×1
Convolutional	128	3×3
Maxpool		$2 \times 2/2$
Convolutional	256	3×3
Convolutional	128	1×1
Convolutional	256	3×3
Maxpool		$2 \times 2/2$
Convolutional	512	3×3
Convolutional	256	1×1
Convolutional	512	3×3
Convolutional	256	1×1
Convolutional	512	3×3
Maxpool		$2 \times 2/2$
Convolutional	1024	3×3
Convolutional	512	1×1
Convolutional	1024	3×3
Convolutional	512	1×1
Convolutional	1024	3×3

Figure 2.11: Darknet-19 architecture

During the forward pass, the architecture performs a total downsampling of 32, so YOLO's original input size of 448x448 is reduced to 416x416 to yield an odd size of 13x13 for the output feature map.

The main components that were added or modified to increase performance concerning the detection task are:

1. Batch normalization: the addition of batch normalization layers after each convolutional layer of the architecture helps reaching convergence during training, without the need of supplementary forms of regularization and allows removing any dropout without incurring in overfitting problems.
2. Introduction of anchor boxes: taking inspiration from successful detectors like RCNN [42], YOLOv2 removes the terminal FC layers and uses the concept of anchor boxes. Anchors are predefined box shapes that are selected to be a good representation of the training dataset annotation shapes. The idea behind anchor boxes is that the majority of objects in a given training dataset and, more in general, in the real world can be grouped in a set of typical height-width ratios. This significantly simplify the training phase, as the detector only has to refine the coordinates of the predictions starting from those of the anchors, instead of starting with random guesses placed anywhere on the image. In addition, prior boxes also naturally enhance the detection at multiple scales, providing improved recall. Instead of hand-picking them, YOLOv2 uses k-means clustering to generate the most suitable anchors for the dataset in use. For VOC and COCO datasets, k = 5 clusters were used.
3. Instead of predicting unconstrained offsets from anchors coordinates like many region proposals based detectors, YOLOv2 keeps following the original idea of directly predicting bounding box coordinates, relative to the location of the grid cell. To obtain bounded predictions a logistic activation is used in the last convolutional layer. The model predicts k bounding boxes per grid cell, where k is the number of anchor boxes used and 5 coordinates (t_x, t_y, t_w, t_h, t_o) plus class scores for each bounding box; a bounding box is also assigned to the anchor sharing the highest IoU with it. For a given prediction the final box coordinates are given by:

$$b_x = \sigma(t_x) + c_x \quad ; \quad b_y = \sigma(t_y) + c_y \quad ; \quad b_w = p_w e^{t_w} \quad ; \quad b_h = p_h e^{t_h}$$

- c_x and c_y are the position of the grid cell on the output feature map that contains the centre of the anchor responsible for the current detection.
- σ is the sigmoid function.
- $\sigma(t_x)$ and $\sigma(t_y)$ the offsets of the centre of the prediction from the grid cell border.
- t_w and t_h control the adjustment of the prediction in respect of the anchor's width and height.
- p_w and p_h are the anchor's width and height.

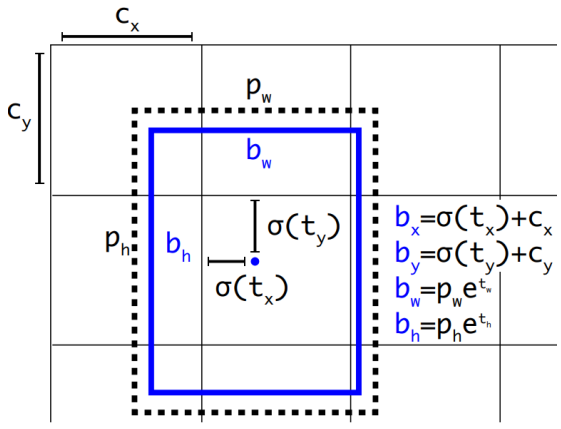


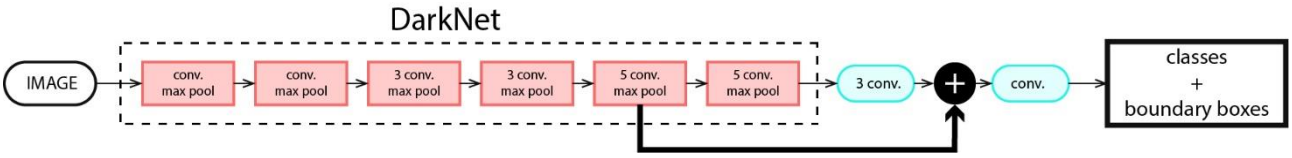
Figure 2.12: Bounding box regression

Finally, the last prediction t_o (objectness score) is used to calculate the box score as:

$$\Pr(object) \cdot IoU(b, object) = \sigma(t_o)$$

4. Multi-scale training: the absence of FC layers enables the architecture to be easily resized during training iterations, enhancing the robustness of the detector to different input image sizes.
5. Use of fine-grained features: the localization of small objects can usually benefit from the integration of higher resolution feature maps, as already proved by SSD detector. To take advantage of this, the second to last layer 26x26x512 feature map is fed into a pass-through layer, where it is reshaped to 13x13x2048 and concatenated to the original 13x13x1024 feature map from the last convolutional layer.

The final concatenated volume is eventually passed to a 1x1 convolutional layer that changes its depth, for a final shape of 13x13x125 encoding the predicted boxes when a standard VOC dataset is used. To sum it all up, YOLOv2 works on input RGB images of size 416x416 and returns a volume with shape SxSx[Kx(B+C)], where S defines the grid cell number (input size divided by 32), K is the number of boxes per grid cell (equal to the number of anchors), B the coordinates defining a box and C the number of classes (20 for VOC). Looking at the numbers, it is clear how YOLOv2 overcame the strict limitation on the number of detectable instances: the original paper implementation can in fact localize up to five objects per grid cell, for a maximum of 845 boxes, although this value can be increased by changing the number of clusters when performing k-means on the training dataset. At native resolution, this version provided performances in line with state-of-the-art at almost 70 FPS, whereas higher resolution inputs outperform it with 78.6% mAP@0.5, still maintaining real time inference.



1Figure 2.13: YOLOv2 full architecture with highlighted passthrough

2.4.6.2 YOLOv3

As already mentioned, YOLOv3 is the last official release by Joseph Redmon. As admittedly stated by its author, this is an all-round update integrating many new valid ideas proposed in the deep learning scenario after the previous release, on top of another new and deeper backbone feature extractor, called Darknet-53 [43]. Unsurprisingly, this architecture is composed by 53 convolutional layers and it merges the approach already followed by Darknet-19 with the introduction of skip connections: it is a concept introduced by He et al. in their paper [44] about the use of residual networks for image recognition, which consists in feeding a layer activation further in the network by adding it to the computation of a successive layer activation. Formally, a basic skip connection can be written as:

$$a^{l+2} = f(z^{l+2} + w_s a^l)$$

where, following the notation already seen in the introductive chapter on neural networks, a is the output of a layer after the activation is applied. F is the activation function of the current layer and

z is the output of a neuron prior to activation. The multiplicative term w_s is a matrix of learnable parameters to make the skip connection feasible even when the two layers have different output size, otherwise the two terms inside the parenthesis could not be summed. This is most often not the case though, as "same" type of convolution is generally used in any residual CNN. The combination of the skip connection and the two layers that it connects is known as a residual block and it is the basic element of any residual network architecture (ResNet). The use of skip connections makes it possible to successfully train very deep networks, reason why the authors decided to exploit them in a very deep architecture like Darknet-53.

	Type	Filters	Size
	Convolutional	32	3 × 3
	Convolutional	64	3 × 3 / 2
1×	Convolutional	32	1 × 1
	Convolutional	64	3 × 3
	Residual		
	Convolutional	128	3 × 3 / 2
2×	Convolutional	64	1 × 1
	Convolutional	128	3 × 3
	Residual		
	Convolutional	256	3 × 3 / 2
8×	Convolutional	128	1 × 1
	Convolutional	256	3 × 3
	Residual		
	Convolutional	512	3 × 3 / 2
8×	Convolutional	256	1 × 1
	Convolutional	512	3 × 3
	Residual		
	Convolutional	1024	3 × 3 / 2
4×	Convolutional	512	1 × 1
	Convolutional	1024	3 × 3
	Residual		

Figure 2.14: YOLOv3 backbone Darknet-53

For performing detection, several other convolutional layers are added to the base feature extractor, for a total of 106 layers. The encoding of predictions is not different from the previous version of YOLO, except for the classification part: this difference is due to the fact that newer datasets such as Open Images features non mutually exclusive classes (for instance, "person" and "woman"). Instead of using a softmax-like approach, where only one class is assigned to a given bounding box, YOLOv3 runs independent logistic classifiers for each class, allowing for multi-label classification. As a consequence, the classification term in the total loss is switched from the previous sum-error formulation to binary cross entropy. The most influential change for YOLOv3 is performing separate detections at three different scales, corresponding to downsampling factors of 32, 16 and 8; as already explained for YOLOv2, the finer-grained features combined with the more abstract ones from deeper maps should help with the detection of small objects on the input image. The input image in the paper description is still 416x416, but it can be adjusted as long as it is a multiple of 32. First detections come from layer 82 in the network, on a 13x13 feature map. Then, the map from two layers prior is upsampled with a factor of 2 and concatenated with the map of a

shallower layer (61) and this 26x26 map is used to perform prediction at the second scale. This exact aggregation process, which adopts ideas behind Feature Pyramid Networks (FPN) [45], is repeated another time to get detections from the third and last scale, where a 52x52 map is outputted. Given the increased complexity of the architecture, a useful schematic is reported below. For YOLOv3, authors started to use COCO dataset to carry out evaluation (which has 80 separate class categories) and in the paper implementation they chose to use three anchor boxes for each scale. As a consequence, any given grid cell predicts three bounding boxes, encoded by using 255 numbers (3 x (5+80)); considering the size of the feature maps used to perform detection at the three scales (13x13, 26x26, 52x52), YOLOv3 can localize up to 10647 instances on the input image, finally make it suitable for applications with a high density of objects. Due the rather deep and more complex backbone and detection head, YOLOV3 traded part of its unrivalled speed for a greater performance return: at standard 416x416 input resolution, it reaches 55.3% mAP@0.5 on the challenging COCO dataset, while still running at the edge of real-time inference and shows comparatively higher improvements in the correct localization of small objects.

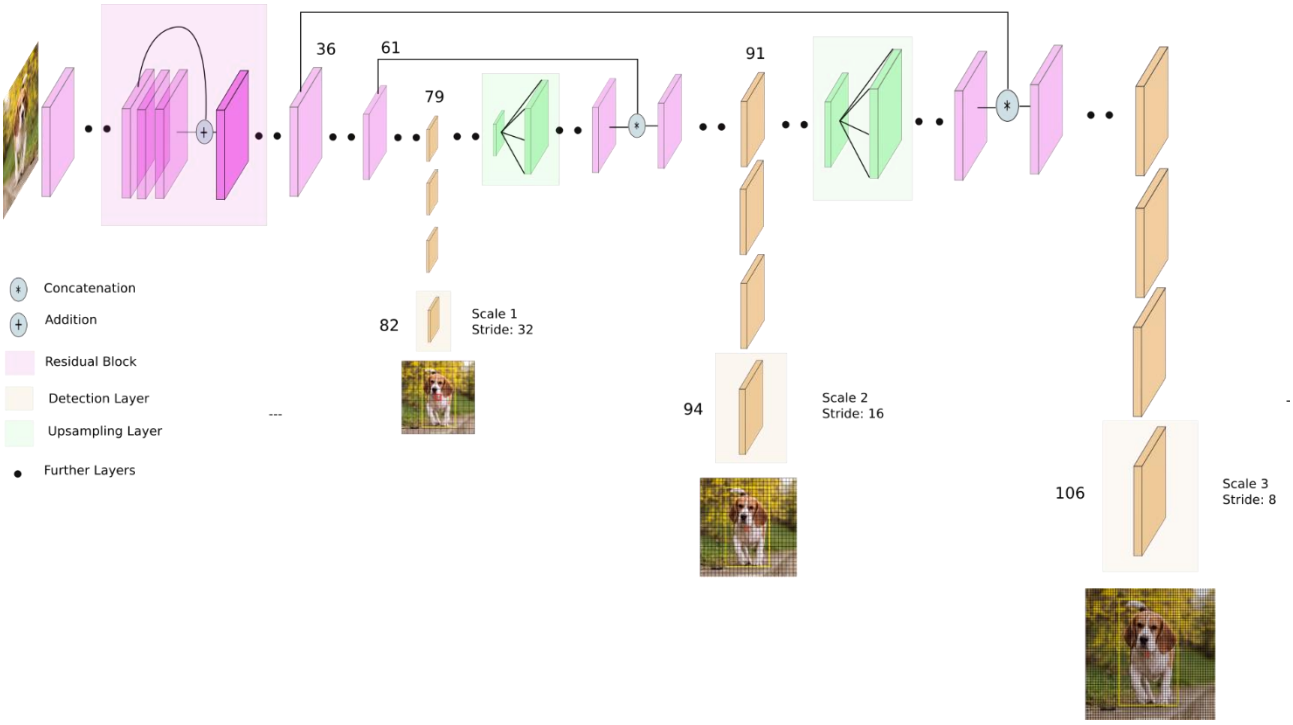


Figure 2.15: YOLOv3 full architecture

2.4.6.3 YOLOv4

YOLOv4 saw its release in the first half of 2020 and, as already hinted in this chapter introduction, it represents the first independent implementation since Joseph Redmon left computer vision research. As authors declare in the first part of their proposal paper, YOLOv4 is the result of extensive experimental evaluation on the impact of various new emerging concepts, which showed promising results in the deep learning field, when they are integrated with the YOLO base structure [46]. As a consequence, YOLOv4 is a fairly complex detector and an in-depth description of all the additions would require much more than a single subparagraph as this one and only the most impactful novelties regarding different aspects of the model will be discussed below. As far as the architecture is concerned, the key modifications are:

- Backbone: YOLOv4 is based on a modified version of the previous Darknet-53 feature extractor, called CSPDarknet-53. The introduction of Cross-stage partial connections (CSP) makes the CNN architecture computationally more efficient, while also increasing its base performance in classification tasks [47].
- Feature aggregation method: as discussed in its subparagraph, YOLOv3 used a concept similar to FPNs to combine the features from different levels in the network. Instead, YOLOv4 relies on a newer approach called PANet [48].
- Spatial Pyramid Pooling (SPP) [49]: this block performs pooling in a different manner than classical pooling layers by outputting a fixed length feature vector irrespective of the size of the input. The introduction of a SPP block after backbone's last convolutional layer increase the robustness of the architecture to input image size variations.
- Mish activations: Mish is a relatively new activation function that proved to outperform ReLu in many experimental settings [50]. Its mathematical definition is:

$$Mish(x) = x \cdot \tanh(\ln(1 + e^x))$$

YOLOv4 doesn't introduce any change concerning the detection head from YOLOv3, so predictions still come from three different scales and their encoding is unchanged. On contrast, new techniques were introduced to improve training process effectiveness:

- Mosaic augmentation: at each iteration, input images in the batch are created as a combination of four separate training images, arranged in random proportions. This increase the model generalization, helping to correctly detect objects even when they are out of their typical context.
- Self-Adversarial Training (SAT): during training, SAT identifies portions in the input images that are particularly critical to the model when taking predictions and alters them in an attempt to reduce this reliance that could possibly undermine the overall model generalization capability.

Figure 2.16: A training example with mosaic augmentation

- DropBlock regularization: it shares a similar aim with SAT, but this regularization method simply obscures a portion of the image directly in the input layer of the model.

Thanks to the integration of the described methods among others, YOLOv4 outperforms the previous version both in terms of inference speed and mAP on COCO, with about 64% mAP@0.5 and 44% mAP@0.5:0.95.

2.4.6.4 YOLOv5

Just a few months after the release of the previous version, another independent team published its contribution on making YOLO faster and more robust. The official paper is still due to be published, while the complete code is open-sourced and already available for use in a dedicated GitHub repository [51]. YOLOv5 doesn't feature a high number of changes to the base architecture and can be seen like a parallel set of different optimizations when compared to YOLOv4 (indeed, their releases are only a few months apart). The backbone is in fact still a CSP based CNN with PANet

feature aggregation and the detection head is not different to that of YOLOv3. Concerning the loss for box regression, YOLOv5 employs Complete IoU loss (CIoU) [52] for the localization term of the total loss, which gives a much more comprehensive context on the correctness of localization, compared to both squared error and classical IoU losses. The latter is simply computed as $L_{IoU} = 1 - IoU$, but such a formulation lacks any awareness on bounding box shape and amounts to zero for any isolated predictions, irrespective of the distance from the nearest GT box; this can also cause slower convergence due to vanishing gradients issues. A first improvement on this matter is obtained by adding a term penalizing distance between the two centres; this is usually called Distance IoU loss.

$$L_{DIoU} = L_{IoU} + \frac{\rho^2(b, b^{gt})}{c^2}$$

In the formulation above, $\rho^2(b, b^{gt})$ is the Euclidean distance between the two bounding boxes centre point and c is the diagonal length of the smallest bounding box enclosing the two boxes. Another factor that can bring useful information about prediction quality is its aspect ratio: a prediction showing inaccurate size, but an aspect ratio that is close to that of the ground-truth should not be penalized as heavily as any other random prediction. Complete IoU loss takes this into consideration by adding a term related to consistency of aspect ratios; this is hence the final formulation that replaces the original localization loss term in YOLOv5.

$$L_{loc} = L_{CIoU} = 1 - IoU + \frac{\rho^2(b, b^{gt})}{c^2} + \alpha v = L_{DIoU} + \alpha v$$

Above, α and v are a positive weighting parameter and the aforementioned consistency factor. They are formulated as:

$$\begin{cases} v = \frac{4}{\pi^2}\left[\text{atan}\left(\frac{w^{gt}}{h^{gt}}\right) - atan\left(\frac{w}{h}\right)\right]^2 \\ \alpha = \frac{v}{(1 - IoU) + v} \end{cases}$$

Considering the training process, this version incorporates Mosaic Augmentation as YOLOv4 already did and uses a fully automatic, k-means based method (named “auto-anchors”) for the definition of optimal anchor boxes (number and aspect ratios) for the dataset in use, whereas all previous versions relied on predefined settings or needed manual tuning. It is also possible to run a generic algorithm optimization to get a set of optimal hyperparameters for the data in use. One step back from YOLOv4 relates to activation functions: the authors found no performance gains when using Mish activations on their model, so they decided to stick to the computationally simpler ReLu activation for all the layers, except the last detection layer which uses sigmoid functions (this is needed to bound predictions, as explained in the YOLOv3 subparagraph). At an input resolution of 640x640, YOLOv5 is capable of reaching 69% mAP@0.5 and 51% mAP@0.5:0.95 on COCO dataset, while running at 80 FPS on a single V100 GPU.

2.4.7 Training of the YOLOv5 model

To implement the object detection step inside the proposed framework, a YOLOv5 detector was chosen, due to its current state-of-the-art performances in general purpose object detection and its

simplicity of use, which renders it easily adaptable to custom tasks without significant tweaks. All the code used is open sourced and is part of the official YOLOv5 GitHub repository by Glenn Jocher. The model is fully implemented in Python, using PyTorch deep learning library and the training process was deployed on the Google Colab environment to take advantage of the provided GPU power and keep training time as low as possible. A total of four increasingly deeper models are available for use, allowing to select the desired compromise of detection accuracy and computational complexity.

Model	Learnable parameters (million)	Floating point operations (billion)	Inference time @ 640 [V100 GPU-CPU] (ms)
YOLOv5s	7.2	16.5	6.4 – 98
YOLOv5m	21.2	49	8.2 – 224
YOLOv5l	46.5	109.1	10.1 – 430
YOLOv5x	86.7	205.7	12.1 - 766

All models are pre-trained on COCO dataset for 300 epochs at an input resolution of 640 pixels. To leverage on the general feature extraction capabilities learned on the thousands images of COCO, transfer learning training was performed, starting from the provided weights (in .pt format): to do so, all the layers' weights were initialized using the pre-trained values to be fine-tuned on the custom dataset images, except for the YOLO detection head, responsible of bounding regression and classification. Given the objective of this step to localize the presence of nuclei in the input image, the problem is configured as a single-class detection task. Input data for training is composed by the patched dataset coming from the pre-processing step described a few paragraphs back. After a series of experimental runs, YOLOv5l model was chosen as the most suitable alternative: smaller architectures yielded, as expected, reduced performance whilst YOLOv5x suffered of early overfitting, probably due to the limited quantity of input data. Anyway, to contain overfitting problems to the minimum, extensive online (directly performed during the training process) augmentation was employed; the transformation applied are summarised with their respective probabilities in the table below.

Transformation	Probability
Mosaic augmentation	1
Translation	0.1
Rotation	0.1
Scale variation	0.3
Shear	0.05
Flipping (up-down/lateral)	0.25

No kind HSV color augmentation is used, as that would just revert the advantages obtained through color normalization. Mosaic augmentation, already explained in YOLOv4 section, was set to a probability of one, so that any input image is actually a pseudo-random mix-up of four patches in the dataset. This is because it proved to be the most influential parameter in reducing generalization problems among those listed. As far as the training hyperparameters are concerned, they were set as follows for the experiments with the most satisfying results:

- Input image resolution: set to the same value of the extracted patches size. For the reasons listed in the YOLOv3 section, input size must be a multiple of 32. If this is not the case the input image is automatically upscaled or downscaled to the nearest acceptable value.
- Epochs: maximum of 350, with an early stopping patience of 100 epochs when performance stops increasing on the validation set. For the purpose of this check, performance value is computed as an adjustable weighted average of precision, recall, mAP@0.5 and mAP@0.5:0.95. The weights of each of those were set to [0, 0.2, 0.8, 0]. The focus is hence on mAP@0.5 and recall, as the aim of this object detection step is to obtain at least a rough localization of the highest possible number of nuclei. On the other hand, very accurate matching of bounding boxes with GT is not as important, because the final segmentation shape will only depend on the quality of the softmax and the characteristics of evolved active contours. In addition, false positives that does not have a correspondence on the softmax can be easily removed during the contours initialization step.
- Optimization algorithm: SGD (optionally, Adam is also available)
- Learning rate: YOLOv5 uses a cosine annealing learning rate scheduler, with a base learning rate of 0.01. With this setting, the learning rate starts at very low values, to avoid altering pretrained weights in the first iterations, when the gradients can have high values, then it increases quickly until reaching a plateau, after which it decays linearly to zero in the remaining final iterations.

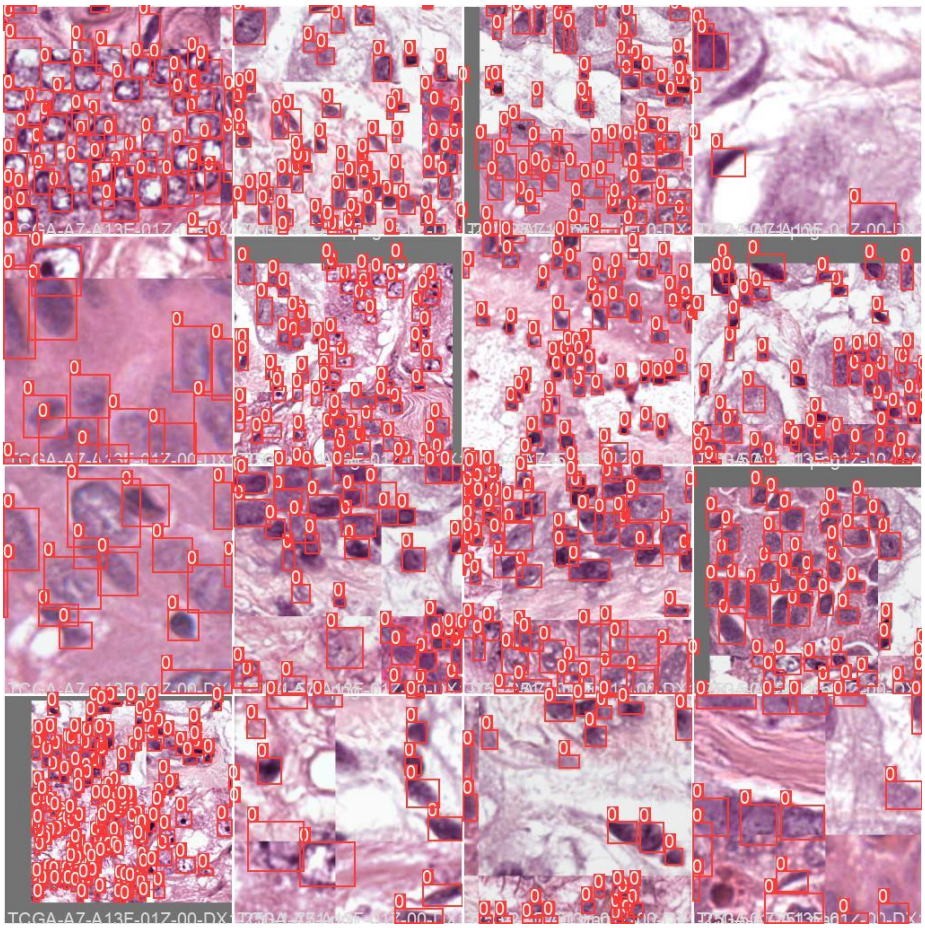


Figure 2.17: A training batch with 250x250 patches (batch size = 16)

After training procedure, YOLOv5 returns the performances on both train and validation data in the form of different plots and metrics. Here are reported the training results for the two input sizes that have shown the best detection quality in the subsequent inference step (640x640, 1000x1000). All the following metrics are computed by considering the typical IoU threshold of 0.5 to define a given prediction as a TP of FP and they represent average metrics calculated only on the single patches, not on the full original image (of course there is no difference for the 1000x1000 case).
One very common and compact way to summarize prediction results for a supervised machine learning method of any kind is the Confusion Matrix (CM), which is a table where columns contain the number of instances of an actual class, while rows represent predicted instances classes (or vice versa, depending on the convention adopted).

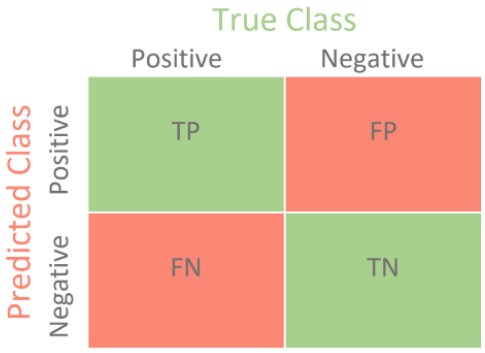


Figure 2.18: A typical confusion matrix

The CMs outputted by the model for the two training runs are reported below. The single-class detection problem does not involve the proper definition of True Negative, as the background is not a class by itself and does not have any associated detection. This is the reason why all the FP are associated with the only used class and the upper right section of any CM is set to unity. All in all, CMs are hence not the best instrument to express performance for object detection tasks, and other methods are usually preferred.

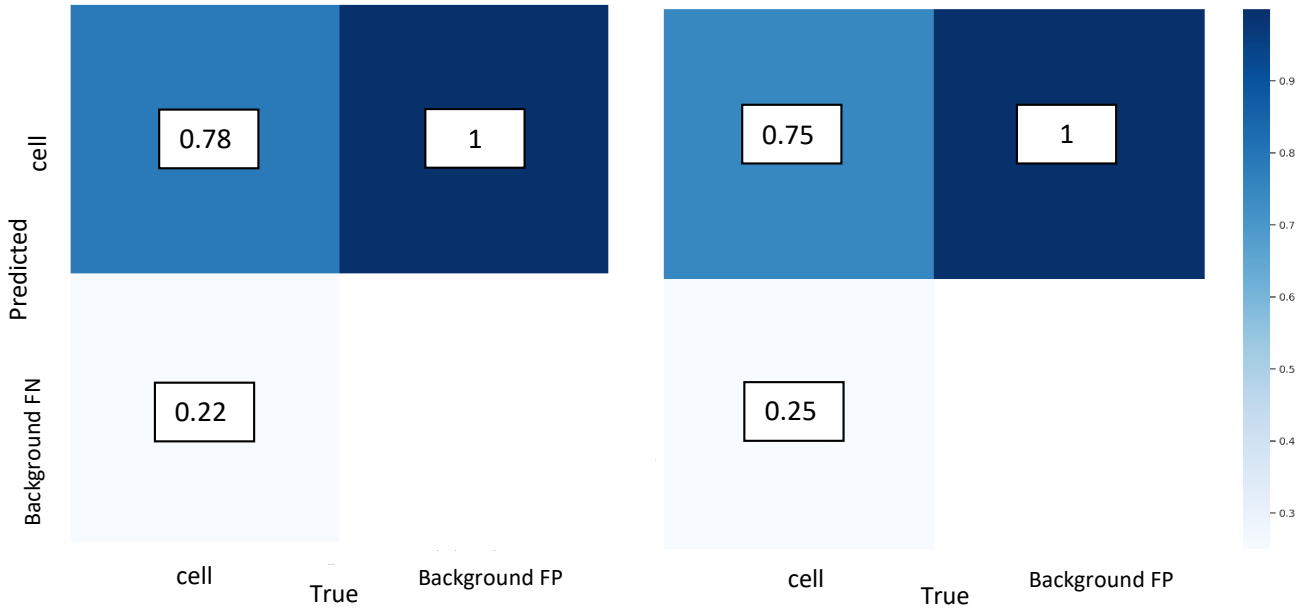


Figure 2.19: CM with 640x640 patches

Figure 2.20: CM with 1000x1000 patches

A more suitable way of determining performance, as discussed in the introduction to object detection, is computing the precision-recall curve, which is then used to compute the relevant mAP metric. In our case of single-class detection, the concepts of AP (AUC of the P-R curve, as previously explained) calculated on the nuclear class and mAP are coincident; cyan and blue lines reported in the legend are indeed perfectly overlapped. 640x640 case shows slightly higher mAP on the patches compared to 1000x1000 case.

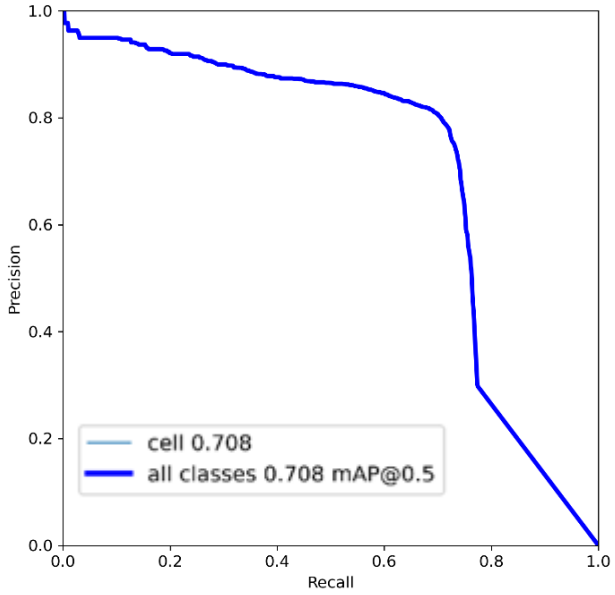


Figure 2.21: P-R curve with 640x640 patches

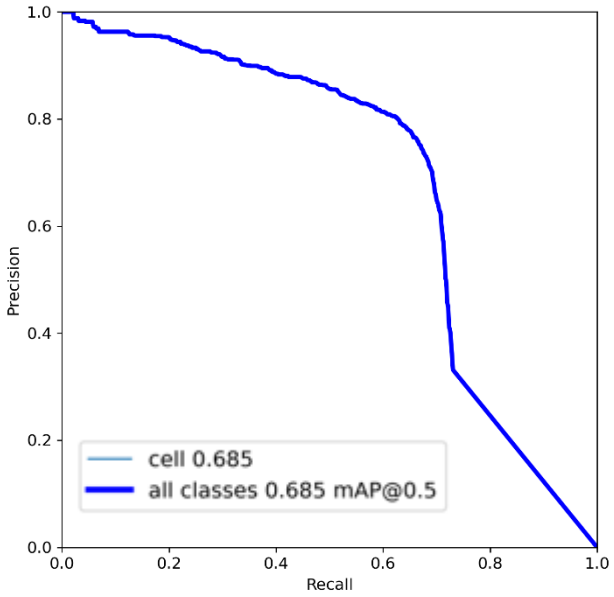


Figure 2.22: P-R curve with 1000x1000 patches

Another insightful plot to consider can be the value of F1-score with the confidence score. This is computed by:

- Taking all predictions in output from the model.
- Applying a threshold on confidence score: all boxes less confident than this value are excluded from the computation.
- Computing the F1-score for the remaining boxes.

This process is repeated several times, increasing the confidence threshold up to the value of 1, and for each repetition a point is added to the plot. The analysis of these graphs can give a hint on a suitable confidence value to choose when performing the successive step of inferencing with the model. For example, a value sligthly above 0.5 could be a good choice to maximize the F1-score of the patch predictions for both models.

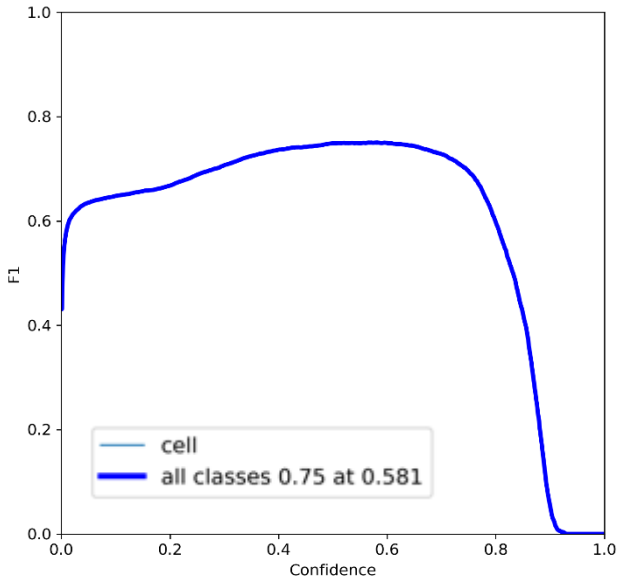


Figure 2.23: F1-score vs confidence (640x640)

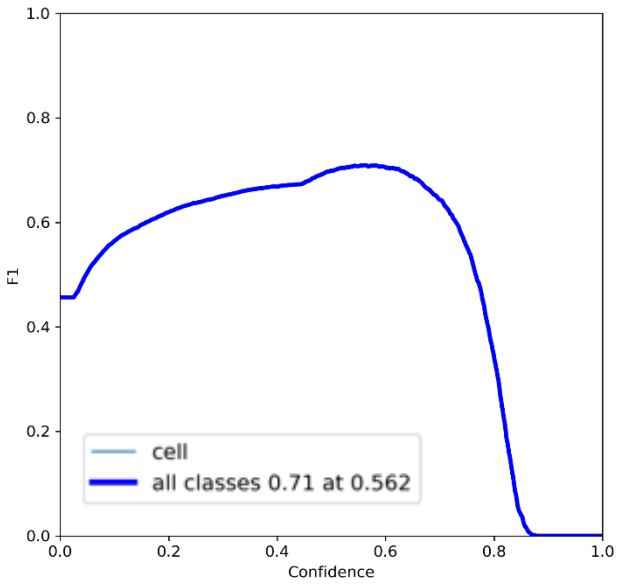


Figure 2.24: F1-score vs confidence (1000x1000)

To complete this analysis on training performance, the actual performance metrics and loss values computed by the network at each epoch is included.

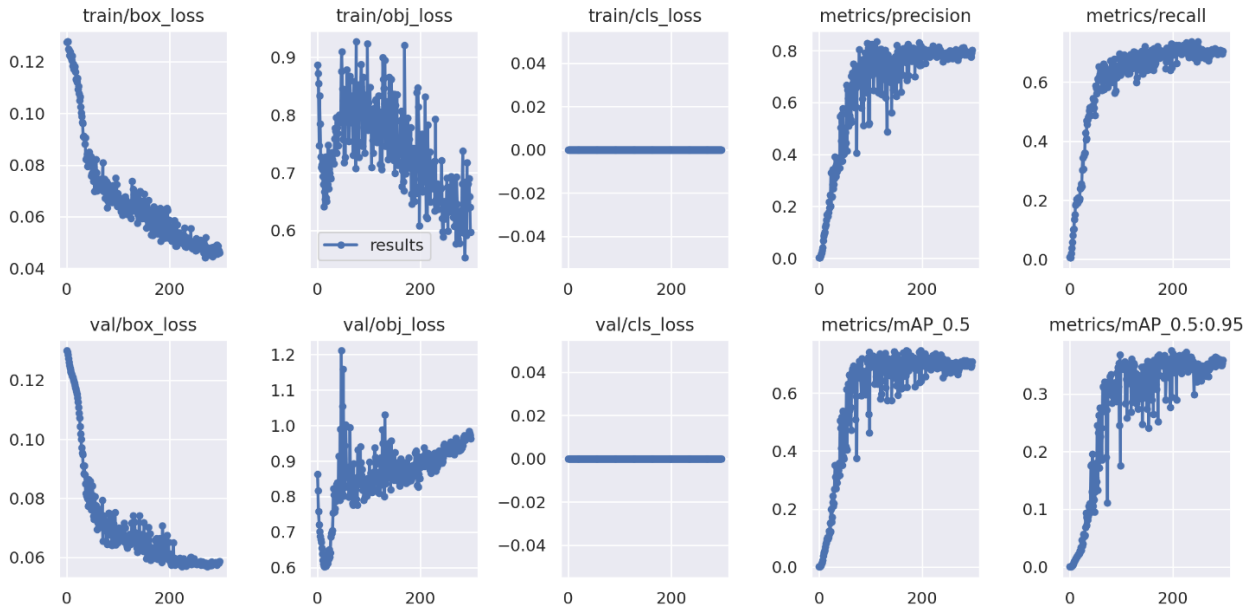


Figure 2.25: Training results (640x640)

The first notable detail is the lack of classification loss. Again, this is due to the simplified single-class nature of this work detection problem: it is important to remember that since its third version, YOLO employs BCE loss, instead of the previous SSE term. The term relative to localization (CIoU) shows a steady decreasing trend, whereas the one relative to objectness has a less clear and oscillating behaviour, in particular on the validation set, where overfitting starts to be apparent after about 150 epochs. Hence, the models show a good capability in localizing the position of objects and optimizing it, while it faces higher challenges, as expected, when trying to tell apart what constitute

an instance rather than just background. However, the various performance metrics, all computed on the validation set, still show a clean increase with the number of epochs, suggesting an effective continuation of the training process.

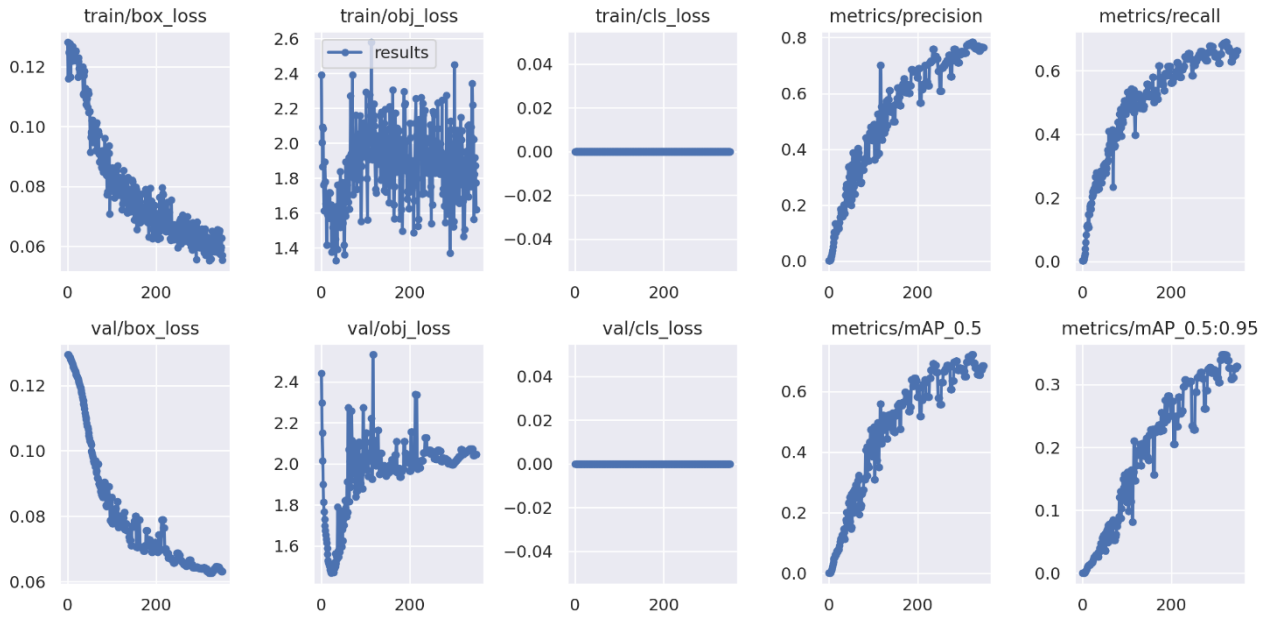


Figure 2.26: Training results (1000x1000)

2.4.8 Model inference and post-processing

Inference of the trained models is performed by inputting all the extracted patches that compose the dataset to the detector and, starting from the patch based detections, reconstructing the full 1000x1000 image. The main parameters of influence in any object detection inferencing process are the minimum confidence score and the IoU threshold for performing NMS. In addition to this, a change was operated to two constants inside the code for the dedicated NMS function ("general.py"), as their default values were too low for this detection problem, leading to performance degradation. These two values are the maximum number of boxes to be given as input to NMS and the maximum number of boxes to be kept after its application: they defaulted to 300 and 1000, which are not sufficient for the high number of nuclei in the input images, and they consequently led to a reduction in recall. The values used for the inference are included in the table below:

	Resolution	
	640x640	1000x1000
Confidence	0.4	0.5
IoU_th	0.35	0.35
Max_NMS (# boxes)	10500	25000
Max_out (# boxes)	1050	2000

The confidence scores used for both models is lower than those corresponding to the maximum of F1 in the plots previously analysed: this is done to maximize detection recall, which is generally more important than precision for the localization of close nuclei, which is the aim of the object detection

step this framework. Moreover, both curves still show a plateau around the selected values of confidence, so F1 is not heavily affected by this choice. With this step, the model outputs separate predictions for each of the patches that compose the initial dataset. Those predictions are then

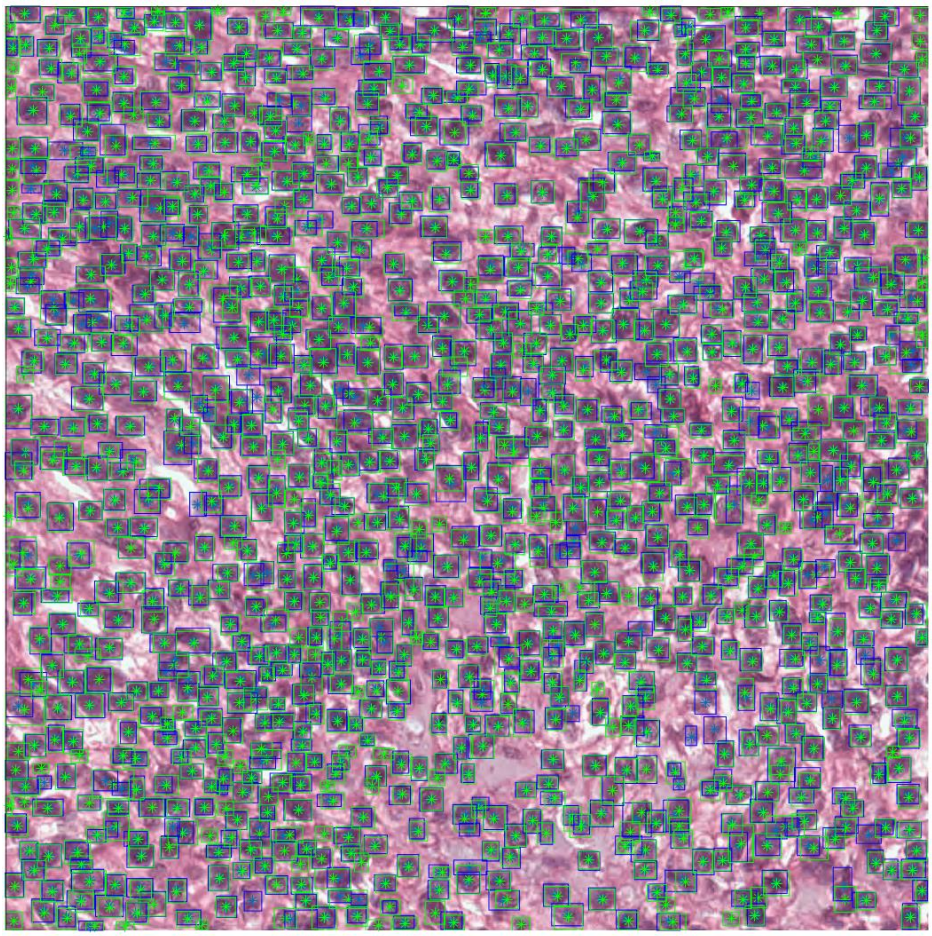


Figure 2.27: Full image detection on a test set example

aggregated together using the information on patch localization saved during their extraction in the preprocessing stage. An additional NMS is applied to all the predictions after their joining, to address for possible double detections due to the overlap of the extracted patches. Finally, a supplementary cleaning of the detections through two ad-hoc developed methods is performed.
The first of those heuristic algorithms serves the purpose of removing residual multiple detections caused by nuclei on the edge of patches: if the extension of such a nucleus on two or more patches is relevant enough, it could easily be detected more than one time by the model. This kind of repeated detection cannot be eliminated by standard NMS, as the bounding boxes predicted from two different patches are completely disjoint. The main steps of this first cleaning method are listed below:

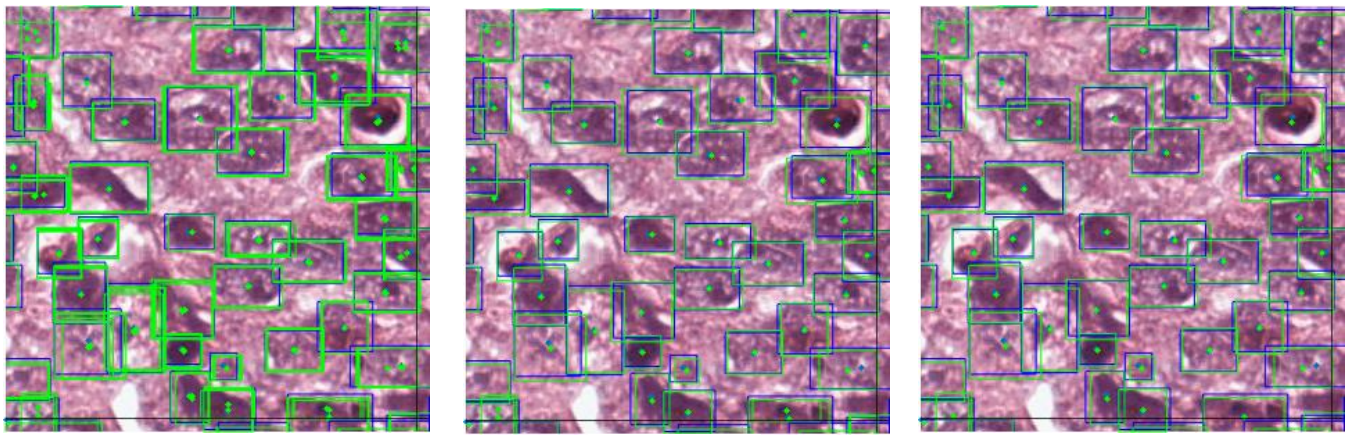
- Predicted bounding boxes are first sorted by area in ascending order. This is to ensure that smaller boxes that are part of a double border detection are removed first, as it is reasonable to suppose that a smaller box is associated with the least representative portion of an object extending over more than one patch.
- Evaluate whether the currently considered box can be considered to be “on the border”: this is the case when at least one of the box coordinates is closer than a threshold to the patch border. A value of 3 pixels was used.

- If the prediction is on the border, search for the presence of another prediction on the same border.
- When a pair is found, compute the Euclidean distance between the pair centroids. If the distance value is lower than a second threshold, remove the first (smaller) box of the pair from the predictions list. A default of 27 pixels was used for this second threshold: a value too high can cause the removal of detections that are on the border, but doesn't represent a double detection, while one too low reduces the effectiveness of the cleaning.

Obviously, the impact of the application of this method is inversely proportional to patch size, as the number of nuclei as the border to area ratio rapidly increase for small patches. The second cleaning step is responsible for the elimination of residual multiple detections that may still be present after the application NMS on the full image. These are mainly due to partially overlapping patches (like in the 640x640 case), when a nucleus is detected more than one time with bounding boxes of significantly different area or aspect ratio (e.g. a small bounding box totally contained inside a bigger one), or when a group of nuclei is correctly identified with their respective boxes, along with a bigger box enclosing them all. In fact, in those cases the IoU value can be low and the NMS algorithm does not consider the presence of a double detection. The main steps for this second algorithm are:

- Predicted bounding boxes are first sorted by area in ascending order. Again, this supposes that smaller bounding boxes contains less information about the detected object than bigger ones.
- Compute the intersection between all bounding boxes (for M predictions, this is a [MxM] symmetric matrix).
- Find the pairs with non-zero intersections, excluding self-intersections.
- If the intersection of a given pair is higher than a consistent portion of the smaller box area (90% is used by default), remove it from the list of predicted boxes.

The use of these two heuristic methods gives reduced, positive benefits to the detection results, with an increment in precision in the face of almost unaltered recall for the 640x640 case and lower patch sizes, whereas their use for the 1000x1000 case is totally irrelevant, since there is no real concept for the images in use.



Pre NMS | Post NMS | NMS + cleaning

Figure 2.28: NMS and heuristic cleaning

2.5 Active Contours

Active contours, also commonly known as deformable models, are a set of image processing and computer vision techniques where a generic profile evolves directly on an image to segment starting from a pre-existing shape, usually by minimising a cost functional. Given this definition, they can be considered as another of the many ways of solving a minimization problem, but considering their formulation, they are naturally suitable and applied in the image domains. Their birth dates back to 1987 with Kass et al. work [53], where for the first time the terms “active contours” and “snakes”. Before proceeding with the description of mathematical formulations and implementation details, its due to introduce a first wide categorisation of these kind of models, which is based on their approach to the problem [54] [55].

- Parametric active contours: also known as “snakes”, the contour is modelled as a curve with fixed parametrization. This is the first approach to develop and it is a direct evolution of original Kass’ work. The main advantages over their counterpart lie in their computational simplicity, ease of control and in the possibility to consider the contemporary evolution of multiple instances and their interaction. On the other hand, their formulation makes them unsuitable for the segmentation of sharp concavities and they show a high sensitivity to noise, initialization (this can undermine the development of fully automatic methods) and image complexity; as a result, they can easily get stuck into local minima and eventually reach suboptimal convergence shapes.
- Geometric active contours: also known as “level-sets”, they were proposed in 1988 [56] to address the main problems encountered when applying snakes. They have a far more complex mathematical formulation compared to parametric models, in which the contour is seen as the zero-level of a curve in a higher order space, as intuitively represented in the picture below. As a consequence of this formulation, they can dynamically change their topology to adapt to any arbitrarily complex shape and they are even suitable for the segmentation of 3D volumes. However, this also renders them computationally heavier and makes it really hard to reframe the evolution process to keep into consideration multiple contours as separate entities.

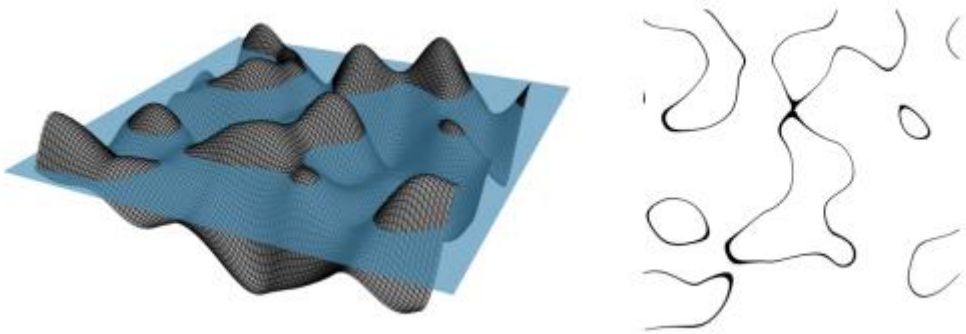


Figure 2.29: A graphical representation of level sets

2.5.1 Parametric active contours

Weighting together the considerations above, which outline the clear advantages and downsides of each approach, with the specific context of this work, the use of parametric active contours was chosen. In fact, the extremely simplified 2D evolution environment offered by a softmax reduces the challenges represented by local minima when snakes are evolved on actual images; moreover, the necessity of handling a great number of contours (one for each detected nucleus) and their mutual interactions makes the choice almost forced.
In the rest of this paragraph, a classical analytical formulation of parametric contours will be exposed [53] [57]. Starting from the representation of the snake in planar curvilinear coordinates,

$$v(s) = [x(s), y(s)]$$

the basic cost functional to minimize is usually written as:

$$E = \int_0^1 \frac{1}{2}[(\alpha|v'(s)|^2 + \beta|v''(s)|^2) + E_{ext}\big(v(s)\big)]ds$$

In Kass' work, E is referred as total energy, while the pair of terms inside the integral are called internal and external energy respectively. Internal energy only depends on the geometrical state of the curve and imposes regularity to the snake: this regularity is controlled by the two coefficients α and β, which weight both derivative terms. In particular, α controls snake's internal tension (curvature term), whilst β is responsible for snake's rigidity (elasticity term). On the other hand, external energy factors in the contribution of experimental data (the image pixels) on the total energy of the curve. As such, it only depends on some characteristics of the image. Which can be different depending on the specific implementation, but in principle it can be considered dependant on the discontinuities of the image and on its local pixel values and it can thus be written as:

$$E_{ext} = w_{line}E_{line} + w_{border}E_{border} = w_{line}I(x,y) - w_{border}|\nabla[k_\sigma * I(x,y)|]$$

The "line" term only depends on the local value of the image and can be used to push the minimum of energy towards bright/dark regions of the image. The "border" term is instead used to create energy minimums across image discontinuities and it is calculated as the gradient of the convolution between the image and a 2D Gaussian kernel with standard deviation σ. w_{line} and w_{border} are simply two weighting constants to balance the contribution of each term on the total energy. Using Euler-Lagrange equation, it is possible to reframe the energy minimization problem into a balance relation between an internal force, which tries to maintain the snake's shape regularity, and an external one that opposes it by adapting the contour to the image shapes.

$$F_{int}(s) + F_{ext}(s) = \alpha v''(s) - \beta v'''(s) - \nabla E_{ext} = 0$$

So in the end, the snake that verifies this relation is also the one that minimizes the energetic cost function and it is the solution obtained after convergence of the algorithm. This basic implementation can be quiet effective, but it suffers from poor convergence problems and it is heavily reliant on a proper contour initialization. Trying to solve these problems, in particular the one linked to initialization, Cohen [58] proposed a model that could also continuously shrink or extend, instead of only evolving from its initial state towards the contours of the image under the only effect of the internal regularizing force. For this purpose, a constant pressure force, oriented

towards the local normal vector of the curve, reason why this formulation is commonly called "balloon snakes". In this case, the total external energy is written as:

$$F_{balloon}(s) = k_1 \vec{n}(s) - k \frac{\nabla P}{||\nabla P||}$$

where k_1 is the amplitude of the inflation/deflation (depending on the constant sign and curve orientation), whereas the second term is the gradient related force that stops the contour. Active contours used for the purpose of this thesis are loosely based on this balloon snakes' implementation, with all the simplifications related to evolving active contours on a probability map, that takes away most of the problems of convergence to suboptimal minima or even divergence.

2.5.2 Contours initialization

After obtaining the final bounding box for each nucleus in the previous detection step, contours are initialized on the softmax probability using each box centroid. All the steps concerning snakes and the subsequent evaluation of segmentation were implemented in Matlab. Detections that have no corresponding segmentation on the softmax are first removed, as they don't have any possibility of evolving, regardless of their correctness (in fact, this unmatching could be both due to a detection false positive, as well as a segmentation network false negative). To do this, all boxes whose centroid lies on a region with a prevalent background probability (red channel), higher than 0.5, are removed and will not generate any contour initialization. Afterwards, contours for the remaining boxes are initialized: to enhance the starting position, instead of directly initializing a contour on the bounding box centre, the centroid of the portion of nuclear segmentation inside the box (obtained by simply thresholding the softmax on a low blue channel value) is computed and the contour is initialized on this point.

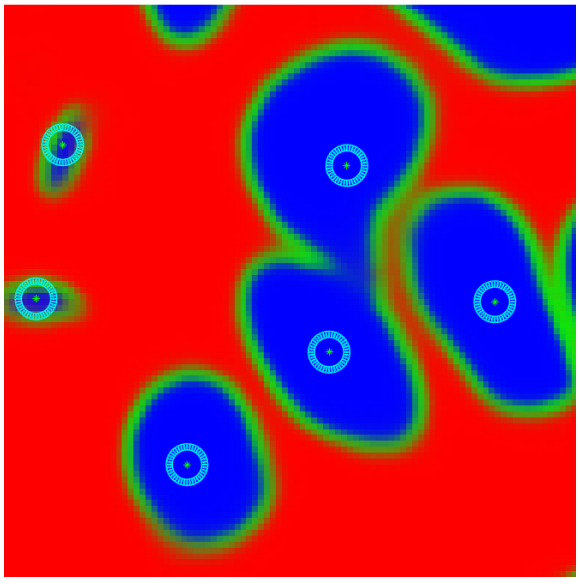


Figure 2.30: Initialized snakes on the softmax map

The main parameters set to define a given contour initialization are listed here:

- Circular shape: this is a simple initialization and it is well related to the actual average nuclear shape.
- Radius: set to 3 pixels, to account for the presence of tiny nuclei on many images of this dataset (as clearly visible in the image below).
- Number of points: 40; this represents a good compromise of computation complexity and shape adaptivity.

Considering the uniform nature of the softmax, initialization should not have a sensible impact on the final segmentation results.

2.5.3 Contours evolution and mask generation

After initialization, snakes are finally evolved on the softmax: the evolution of all contours in a given iteration is treated simultaneously, to exploit vectorization of the code and increase performance. Evolution loop is controlled by these two stopping criteria:

1. All the snakes on the image can be considered as successfully converged, using a specific convergence criterion based on the average evolution rate in the last few iterations. In particular, when the average movement of all N points composing a given contour in the last M iterations is lower than a minimum threshold, the contour is frozen in its current state and is not evolved in the remaining iterations. For the computation of the criterion, M was chosen equal to 40 iterations, while the threshold was set to a value of 0.3 pixels.
2. The maximum number Nmax of overall iterations, set to 1500, is reached. Unless the oscillations due to the mutual interaction between contours are high for at least a contour on the image, this criterion should normally not be triggered.

The evolution of contours at each step is regulated by four coordinates updating rules, two of which are modulated by local softmax probability values. These terms are now listed and their role in controlling the contour evolution is explained.

- Expansion contribute (balloon): this term controls the radial inflation of the contour. The only difference with a classical term of this kind is that it is modulated on the local nuclear (blue) probability, so that the expansion can spontaneously be reduced as the contour approaches the border. It is formulated as:

$$x_{new} = \alpha_1 \cdot p_{nuc}(x) \cdot \frac{x - x_b}{\max(x - x_b)}$$

term controls the radial inflation of the contour. It is modulated on the local nuclear (blue) probability, where p_{nuc} is the probability of classification as a nucleus of the pixel at the actual coordinate and x_b is the barycentre of the contour $x_b = {\sum_{i=1}^{N} x_i}/{N}$.

- Smoothness contribute: this term tries to maintain a regular shape for the contour during evolution. A such, it only depends on the internal snake coordinates at the current iteration:

$$x_{new} = \alpha_2[\frac{1}{2}(x_{left} + x_{right}) - x]$$

where x_{left} and x_{right} are the coordinates of the previous and following contour points in respect of the currently considered one.

- Border contribute: it acts as a typical "line" term, that attracts contour points to high green intensity values (nuclear border probability). This term only depends on the local softmax values, like the typical external energy term.

$$x_{new} = \beta_1[p_{border}(x+1) - p_{border}(x-1)]$$

Here, p_{border} is the probability of classification of the pixel as a nuclear boundary.

- Mutual interaction contribute: this last term regulates the repulsive interactions between two or more interacting snakes, so that segmentation of previously fused nuclei can properly stop. Two given snakes are considered as interacting when at least one of their points intersect the other contour. Any point that interacts with a second contour is pushed back towards the centre of its own contour, following the relation:

$$x_{new} = \beta_2 \cdot \frac{1}{d}$$

where $d = ||x - x'||$ is the Euclidean distance between the considered point and the centroid of the interacting contour.

a_1, a_2, β_1 and β_2 are constant weighting parameters that set the relative impact of each of those term in controlling contours evolution. Again, to the simplified nature of the confronted evolution problem, the set of values chosen for these constants cannot cause divergence of the algorithm, although their tuning can be useful to reach quicker and slightly better overall shape convergence. The values used for the evolution of the object detection initialized contours, for both 640x640 and 1000x1000 cases are indicated in the dedicated table below.

Parameter	Value
a_1 (expansions)	0.1
a_2 (smoothness)	0.05
β_1 (nuclear borders)	0.2
β_2 (repulsive interaction)	1

The snake evolution process performed on the whole dataset takes an average time of 38.9 ± 5 seconds per image on an Intel i3-9100F CPU, with images with a high number of close nuclei taking the most time due to the high number of interactions to be handled. In fact, at the actual state, the function responsible for the computation of the mutual interaction term makes up for almost half of the entire evolution runtime and it is hence the critical part to optimize performance-wise to achieve a first speed up of the whole segmentation framework.

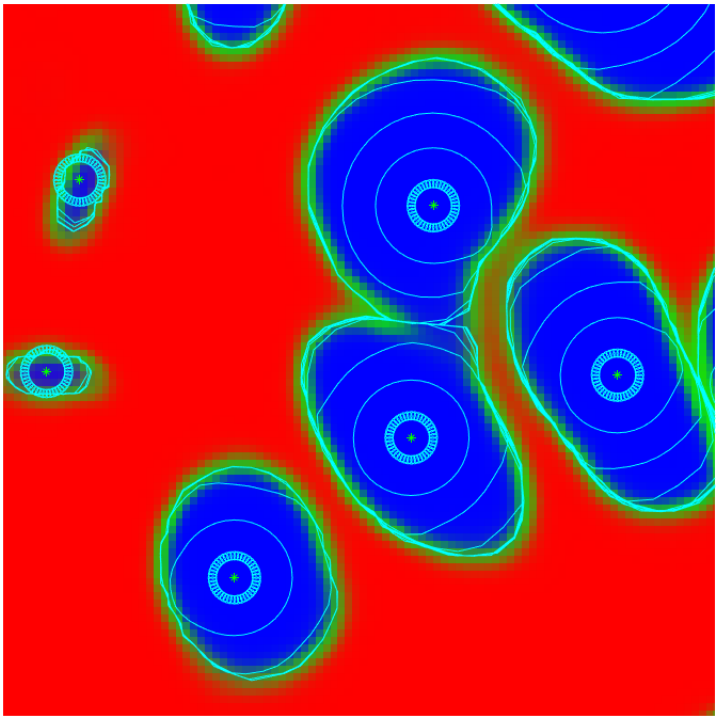


Figure 2.31: Evolution of snakes through the iterations

As a last step to obtain the final image segmentation, the binary mask is generate starting from the fully evolved snakes. Each contour is considered on its own, so that a per instance segmentation that enables a subsequent easy and correct evaluation is possible. To generate its associated mask a built-in Matlab function is employed, which simply consider each contour as a closed polygon composed by N points.

2.6 Validation metrics

In this chapter, the main metrics used to evaluate the yielded results, for both the detection and contour segmentation steps, will be listed and briefly described. The metrics for object detection here described have nothing to do with the ones returned by the model code during training, as those were patch-based as opposed to the full image context here considered. When a particular choice is made regarding the way a metric is computed, it will be justified in the relevant subchapter.

2.6.1 Detection metrics

The final quality of the detection step has been evaluated on the cleaned predictions on the full image, using three typical metrics used for classification tasks: precision, recall and F1-score. In the context of the considered problem, recall is particularly important as it scores the ability of the model in correctly identifying the most part of the GT nuclei on the input image. A low recall on the detection step can indeed offset all the benefits given by the separate segmentation of fused objects, as any missed nucleus won't have an associated segmentation on the final output, even when the segmentation network correctly identified it on its softmax. On the other hand, precision is associated to the model ability at correctly suppressing background boxes; it is less critical than recall in our framework, as false positives that does not have a corresponding softmax match are

easily eliminated during snake's initialization step. The only detrimental false positives are those coming from double detections of the same object, as this is much harder to identify and remove and it leads to an overestimation of the number of instances on the image and, consequently, to a fragmented segmentation after snakes are evolved.

$$precision = \frac{TP}{TP + FP} \quad ; \quad recall = \frac{TP}{TP + FN}$$

F1-score gives instead a rounder estimation of the overall detection quality, without however giving insights on the main causes of a possibly low performance value. It is computed as the harmonic mean of precision and recall:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} = \frac{TP}{TP + \frac{1}{2}(FP + FN)}$$

To compute the metrics for an image, each predicted box is compared to all remaining GT boxes: if a match is found, the predicted box added to the TPs counter and the corresponding annotation is removed from the GT list, while in case of no matching it is added to the FPs counter. The computation stops when all predictions have been considered and the number of residual unmatched annotation boxes is used to get the FNs value. In the computation of all the metrics in this section, an intersection criterion was used in place of the more typical IoU, to decide whether a given predicted box is to be considered a true positive rather than a false negative. To explain the reason for this choice, two effective graphical examples are analysed. Intersection was computed as:

$$Inter = \frac{B \cap B^{GT}}{\min(area(B), area(B^{GT}))}$$

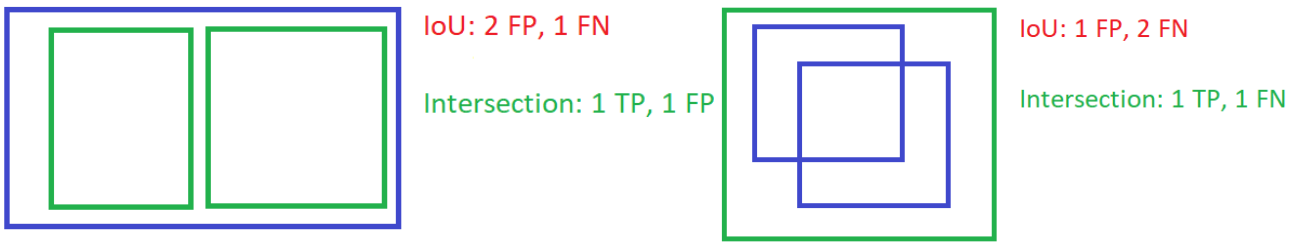


Figure 2.32: Intersection vs IoU comparison

In the image above, blue boxes represent GT annotations and green boxes are possible boxes predicted by the object detection model. In the case on the left, a single nucleus is erroneously detected as two adjacent smaller nuclei by the model. In this case, the use of typical IoU with a 0.5 threshold would result in considering both the predictions as FP, with a resulting FN associated to the GT box, because the value of intersection of both the predicted boxes with the GT is reduced compared to the union area. On the other hand, by using intersection both the boxes have unitary intersection and hence one will be considered as a TP and the other a FP. While this is usually an unwanted behaviour in general object detection tasks, it makes for a better evaluation for the considered matter, as the GT object will have its segmentation in the final output, considering that

the centre of the green boxes still lies inside the object. On the right a similar case is observable, where two close nuclei are considered as one and enclosed together in a bigger box. Of course this is not a good scenario, as the primary aim of the detector is to allow the segmentation of fused objects, but it is more reasonable to consider the detections as one TP and one FN, as the final segmentation will present one single fused object. From these consideration, all the metrics listed above are computed using an intersection threshold of 0.7.

2.6.2 Segmentation metrics

All segmentation results, including the raw segmentation directly computed from the softmax are evaluated on a wide range of metrics. An ideal evaluation criterion used for nuclear segmentation should take into proper consideration many characteristics of the result, most importantly [24]:

1. Completely missed detection of actual objects
2. Erroneous detection of background
3. Under-segmentation
4. Over-segmentation

It's not easy to formulate a single criterion to include all aspects that can affect a given segmentation, as those can even be different from an application to another. For this reason, pixel-level and object-level metrics will be both considered, to assess the benefits of the proposed algorithm on different potential applications. Additionally, a comprehensive novel metric for nuclear segmentation evaluation, Aggregated Jaccard Index, will be detailed. When the relevant formulation of a metric has already been given in the previous chapter, only its role inside the evaluation process will be reported.

2.6.2.1 Pixel-based metrics

Pixel-based performance metrics are calculated by considering each pixel on the predicted mask as a TP or FP on its own. As such, all pixel metrics are computed by simply summing together all the pixels of a kind and using those values to calculate common classification metrics (or in alternative, by applying logical operations directly on the predicted and GT binary masks). Pixel-based methods give a good estimation on the capabilities of the segmentation algorithm in outputting accurate predictions in terms of shape and size, but lacking any concept of instance, they may not be suitable for applications where an accurate estimation of the number of objects is central. The list of the pixel-based metrics used in the evaluation of this thesis results is:

- Accuracy: it is one of the simplest metrics for any classification task. It gives a first estimation of performance, but lacks any further information on the segmentation quality (under/over-segmentation, shape…). It is computed as the ratio of all correctly identified pixels and the total pixel count on the image.

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

- Precision: it measures the ability of the method to avoid over-segmentation.
- Recall: as opposed to precision, it can be used to evaluate under-segmentation issues.

- F1-score: it is computed as the harmonic mean of precision and recall and as such gives a more general view on performance quality like accuracy, but it is better suited when class imbalance is considered.
- Jaccard index: it is a good metric to evaluate shape concordance of the segmentation with the ground truth. It is closely related to the IoU value typically used for object detection, as it is given by the area of intersection of the predicted and GT masks divided by their union area, or it can alternatively be formulated in terms of TP, FP and FN as:

$$Jaccard = \frac{|P \cap GT|}{|P \cup GT|} = \frac{|P \cap GT|}{|P| + |GT| - |P \cap GT|} = \frac{TP}{TP + FP + FN}$$

- Dice similiarity coefficient (DSC) = this metric mostly gives the same kind of information of Jaccard and evaluates the overall overlap between the predicted segmentation and the annotated one. The only difference in computation with Jaccard is that it considers twice the intersection value at the numerator, instead of substracting it from the denominator.

$$DSC = \frac{2|P \cap GT|}{|P| + |GT|} = \frac{2J}{1 + J} = \frac{2TP}{2TP + FP + FN}$$

2.6.2.2 Object-based metrics

The computation of instance-based metrics prerequires the definition of a criterion that makes it possible to define whether a given GT object can be associated with any of the segmented objects, obtaining a counter for FPs, TPs and FNs. All reported indicators define a ground-truth as matched if it is overlapped for more than 50% of its area with a predicted box, as in []. This approach has a better generality in respect of the mask size compared to other common distance-based criteria[]. As opposed to pixel metrics, object-based evaluation is used to quantify the capability of the segmentation algorithm in reliably detecting the presence of an instance in any region of the input image, but it gives limited information, if any (depending on the specific metrics employed), on the morphological quality of that detection. The list of the object-based metrics used in the evaluation of this thesis results is:

- F1-score: in this case, F1 is computed using the already introduced formula, directly using the number of correctly identified GT/predicted boxes and it consequently does not contain information at pixel-level of any kind. It is however a good indicator if the application considered requires an accurate estimate of separate instances that make the image up.
- Jaccard index: in this case Jaccard is calculated on each matching GT/predicted instance pair and the final value is obtained as the average over each considered object. As such, this object-based metric mixes detection quality with average pixel level information about the extent of the overlap between the two masks, typical of Jaccard.
- DSC: the same considerations expressed for Jaccard apply.

- Hausdorff distance (HD): HD is another common metric, often used in problems of shape matching. Given two set of points A and B, one for the predicted mask contour and the other associated to the GT, HD between A and B is computed by first finding the minimum distance from B of each point of A. Afterwards HD is defined as the maximum value among those. The use of Jaccard or DSC is usually preferred as the computational complexity of HD is higher.

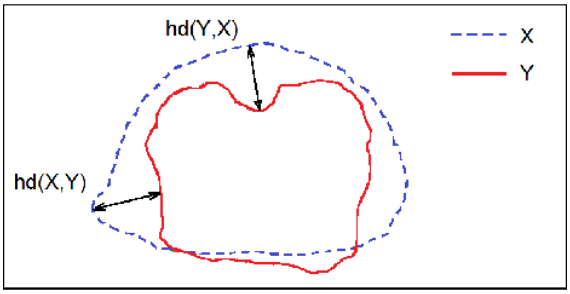


Figure 2.33: Hausdorff distance of two curves

2.6.2.3 Aggregated Jaccard Index

As already hinted in the introductive chapter, in the analysis of histopathological images a great number of heterogeneous factors are weighted by expert clinicians to take their decisions. These include both morphological considerations like shape and distribution as well as more quantitative measurements such as number or relative localization. As a consequence, focusing solely on one type of evaluation metric, be it pixel-based or object-based, can lead to misleading interpretation of the results or incomplete evaluation. To address this problematic, the focus during evaluation will be shifted on a supplementary performance indicator, named Aggregated Jaccard Index. AJI, formulated by Kumar et. al in the same proposal paper of the dataset used in this work [24], is a more comprehensive and general metric specifically thought for the evaluation of nuclear segmentation techniques. AJI is a particularly strict indicator, as it jointly penalizes both object-level and pixel-level errors, effectively bringing together segmentation with detection. It is a modification of the traditional Jaccard index, where, instead of the simple intersection and union of predicted and GT masks, two different pixel sets are used. The main steps towards computing AJI are now explained:

1. First, initialize a correct pixels counter C and a union pixels counter U.
2. For each ground truth annotation, compute the Jaccard index with each segmented object.
3. If the Jaccard is non-zero with at least one prediction, find the pair with the highest Jaccard.
4. Update C and U by adding all intersection pixels to the first counter and all union pixels to the second. Then remove the used shape from the list of predictions and segmentation mask.
5. Otherwise, when GT is not matched to any shape, add all its pixels to U counter.
6. Repeat point 2 to 5 until all GT annotations have been considered.
7. Add the residual pixels on the predicted mask to U.
8. Compute the final AJI value by taking the ratio of C and U.

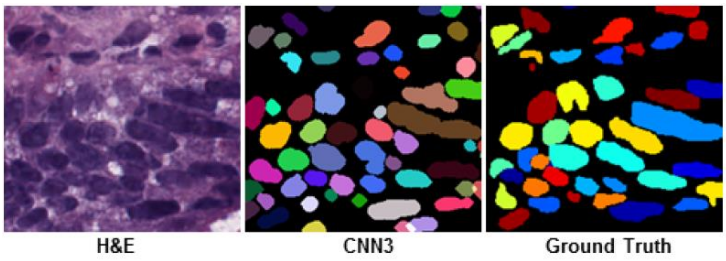
Pseudocode from the original paper is included for further clarification of the described steps.

Input: A set of images with a combined set of annotated nuclei G_i indexed by i, and a segmented set of nuclei S_k indexed by k.
Output: Aggregated Jaccard Index A.

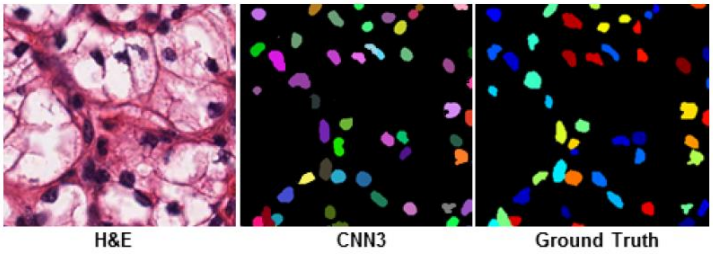
Initialize overall correct and union pixel counts: $C \leftarrow 0; U \leftarrow 0$
for Each ground truth nucleus G_i **do**
    $j \leftarrow \arg\max_k(|G_i \cap S_k|/|G_i \cup S_k|)$
    Update pixel counts: $C \leftarrow C + |G_i \cap S_j|; U \leftarrow U + |G_i \cup S_j|$
    Mark S_j *used*
end for
for Each segmented nucleus S_j **do**
    If S_k is not *used* then $U \leftarrow U + |S_k|$
end for
$A \leftarrow C/U$

Figure 2,34: AJI pseudocode

Thanks to its mixed approach, AJI is capable of penalizing whole missed and ghost predictions, as well as pixel-wise under-segmentation and over-segmentation. In particular, the steps above show how AJI can rapidly decrease as the number of missed or fused objects increase: in those cases, in fact, all the pixels from the missed GT, along with the eventual over-segmentation part of the fused detection are added to the denominator in the calculation. For this reason, the proposed framework should carry a sensible advantage in terms of this indicator when applied to any baseline softmax segmentation.



(b) Colon: AJI=.49, F1S=.71, DC=.77, HD=7.1



(a) Kidney: AJI=.67, F1S=.71, DC=.78, HD=6.9

Figure 2.35: AJI comparison to traditional metrics

The image above, directly taken from the AJI proposal paper, shows how two images with an evident segmentation quality discrepancy may actually feature similar performance value as far as both pixel-level (F1-score) and instance-level (HD) performances are considered individually, whereas their evaluation through AJI correctly express this quality difference.

Results

In this chapter the most meaningful experimental data that can be useful to characterize the proposed framework general performance, advantages and criticalities are reported. All the results have been computed according to formulations and steps reported at the end of the Materials and methods chapter. All the results are relative to the use of patches of size 640x640 and equal to the whole 1000x1000 image, as they yielded the best performance through the various steps. This might be unexpected at first, considering that smaller patch sizes provide the detection model with more training examples (the total number of input images are 52 for the first case and just 13 for the latter) and reduce the number of detections to be outputted per image; however this can be due to the fact that all YOLOv5 models are pre-trained at exactly 640 square input resolution, and the features learned by the backbone may hence perform better around those value of resolution and above.

3.1 Detection performance

First, the performance of the object detection model on the full images is reported. It is crucial to make sure that the values obtained at this stage are acceptable before selecting a model to use for the subsequent initialization of the snakes, otherwise the whole segmentation process and the main idea behind the developed framework will be hindered by the poor detection performance. The bar graphs that follow summarize the average performance in the three detection metrics, which were computed using the ideas in paragraph 2.6.1, on all the four splits of the dataset.

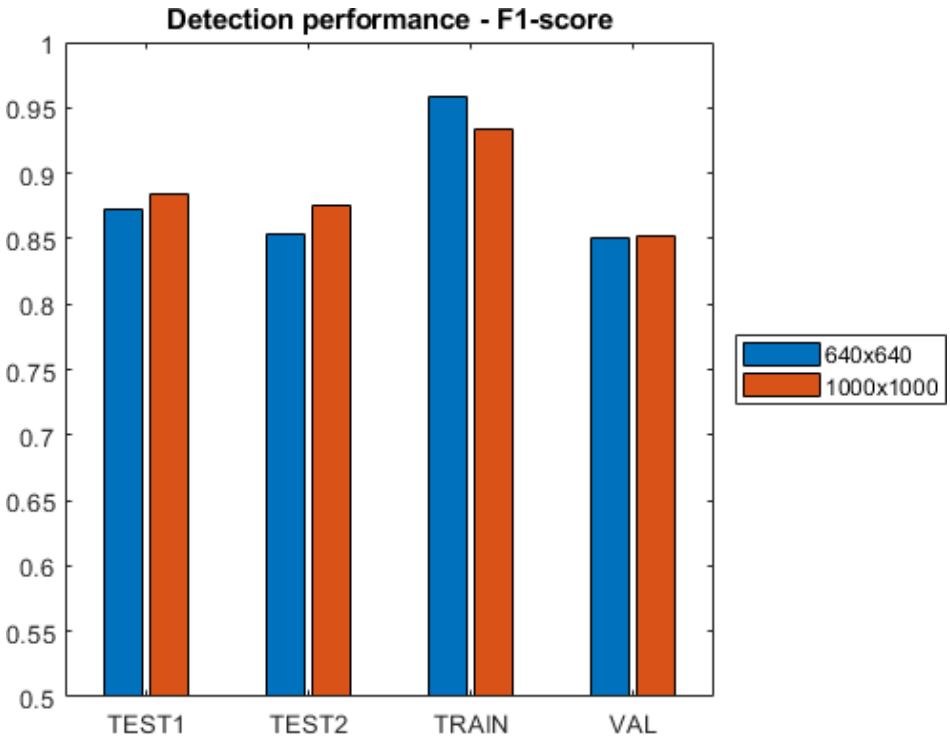


Figure 3.1: F1-score computed on the bounding boxes

The graph above shows good F1-score values, always above 0.85. This suggests that both the models are likely to bring a useful contribution when later used to initialize the active contours on the softmax probability map. There is a non-negligible performance gap when comparing performance on the training and validation set, which suggests that a certain degree of overfitting is still present, in spite of the attempts to contain it (e.g. through the extensive data augmentation applied). However, the results on both the test sets are a confirmation of the good generality and representativeness of the validation set used to monitor the training process, as they feature performance in alignment with or even better than it, especially for the 1000x1000 model. In general, the detector trained with 1000x1000 images boasts slightly higher and more uniform values of F1-score across the four sets and it is thus less affected by the aforementioned overfitting tendency.

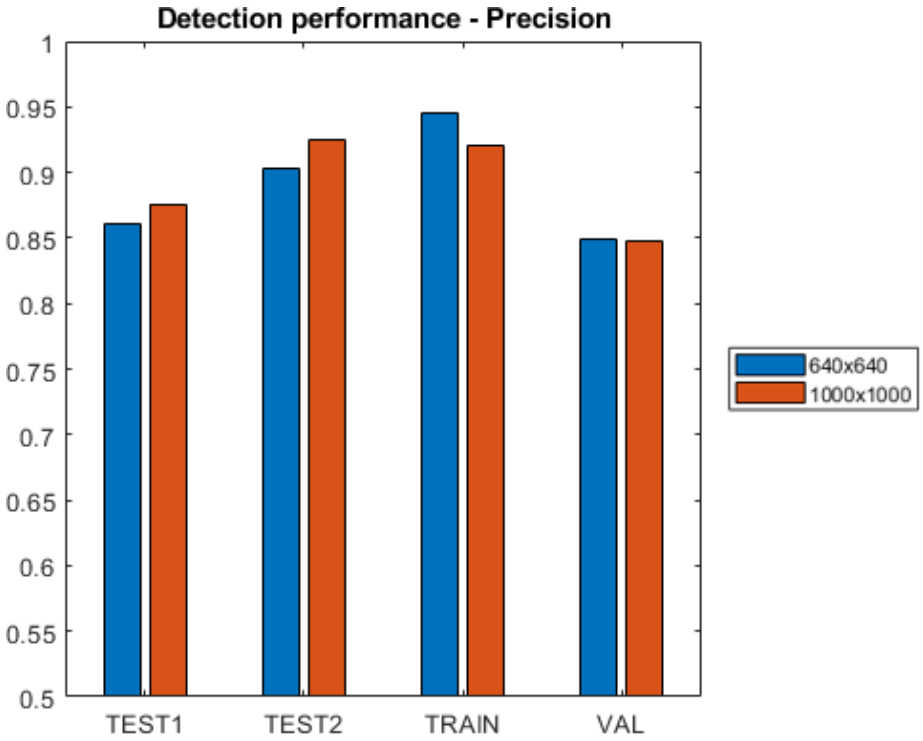


Figure 3.2: Precision computed on the bounding boxes

The precision metric is perhaps the least insightful when trying to define whether the trained object detection model is good enough to be carried over in the framework. In fact, most of the false positives are either eliminated during initialization as explained in chapter 2.5.2 or were already erroneously segmented by the segmentation network itself and as such they should not represent a source of performance degradation. As already noted, the only kind of FP that can weaken the purpose of the object detection model are multiple detections, but the aggregated precision metric does not allow to differentiate among those cases. For both detectors, the lowest value of precision is associated with the validation set and it is higher for both the test sets.

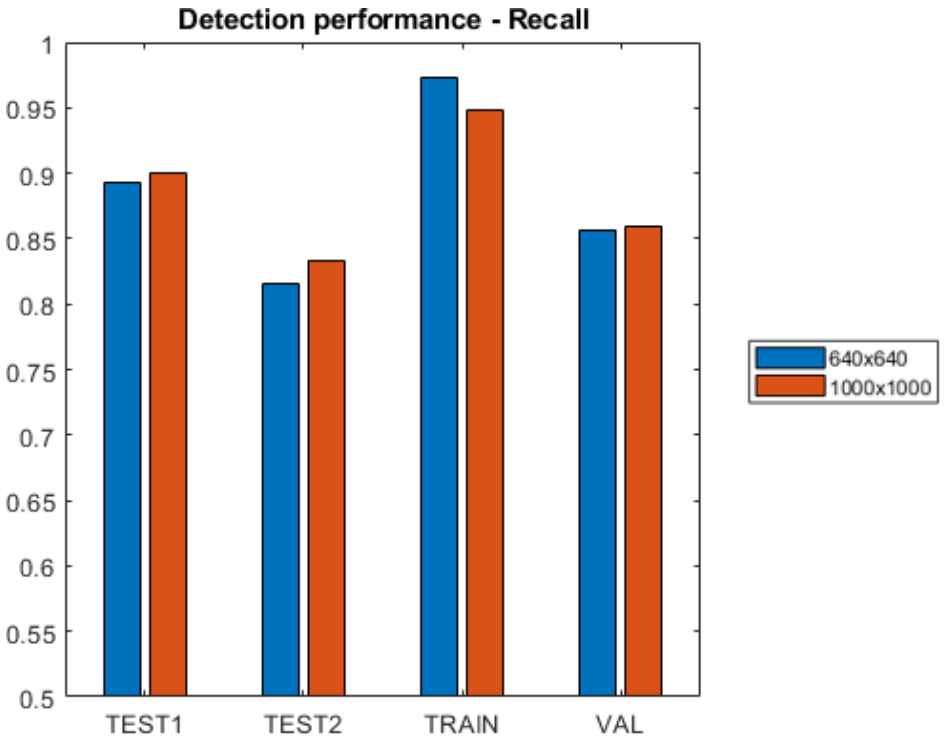


Figure 3.3: Recall computed on the bounding boxes

The average values of recall show an acceptable capability of the detectors in localizing a good part of the nuclei on the dataset images. Recall values on train, validation and test 1 are all higher than those of precision, which is an ideal situation. Unfortunately, in this case the lowest performance is found on the second test set; this is unsurprising, considering that it contains tissues from organs that the model never saw during training, even though such a low value, especially for the 640x640 case, may start to significantly reduce the benefits given by the introduction of the object detection model in the pipeline. Considering however that test 2 have fairly high precision values (even the highest one among all sets for the 1000x1000 model), there is room for improvement if some of its precision is traded in for a comparable or slightly lower recall increase.

3.2 Final segmentation performance

All the segmentation metrics here reported have been computed following the considerations discussed in paragraph 2.6.2. The baseline performance values have been obtained by applying a simple thresholding segmentation on the softmax, following an approach commonly used by deep learning frameworks without a post-processing step. In particular, the steps to generate the raw binary mask are:

1. Apply a thresholding on the softmax green channel (nuclear border probability) with a threshold value of 0.35.
2. Apply a thresholding on the softmax blue channel (nuclear “inside” probability) with a threshold value of 0.5.
3. Substract the border mask from the “inside” nuclear mask.
4. Remove negative values from the final logical mask.

For compactness and better visualization purposes, in all the evaluations presented in this section the validation set was merged with the training set; its explicit presence was anyway more important for detection evaluation, as it had a specific role during the training phase of the models. First of all, the pixel-based segmentation metrics are reported and discussed. While they do not represent the main focus of this evaluation, they can be useful to start understanding the effects of the proposed framework on the general segmentation quality.

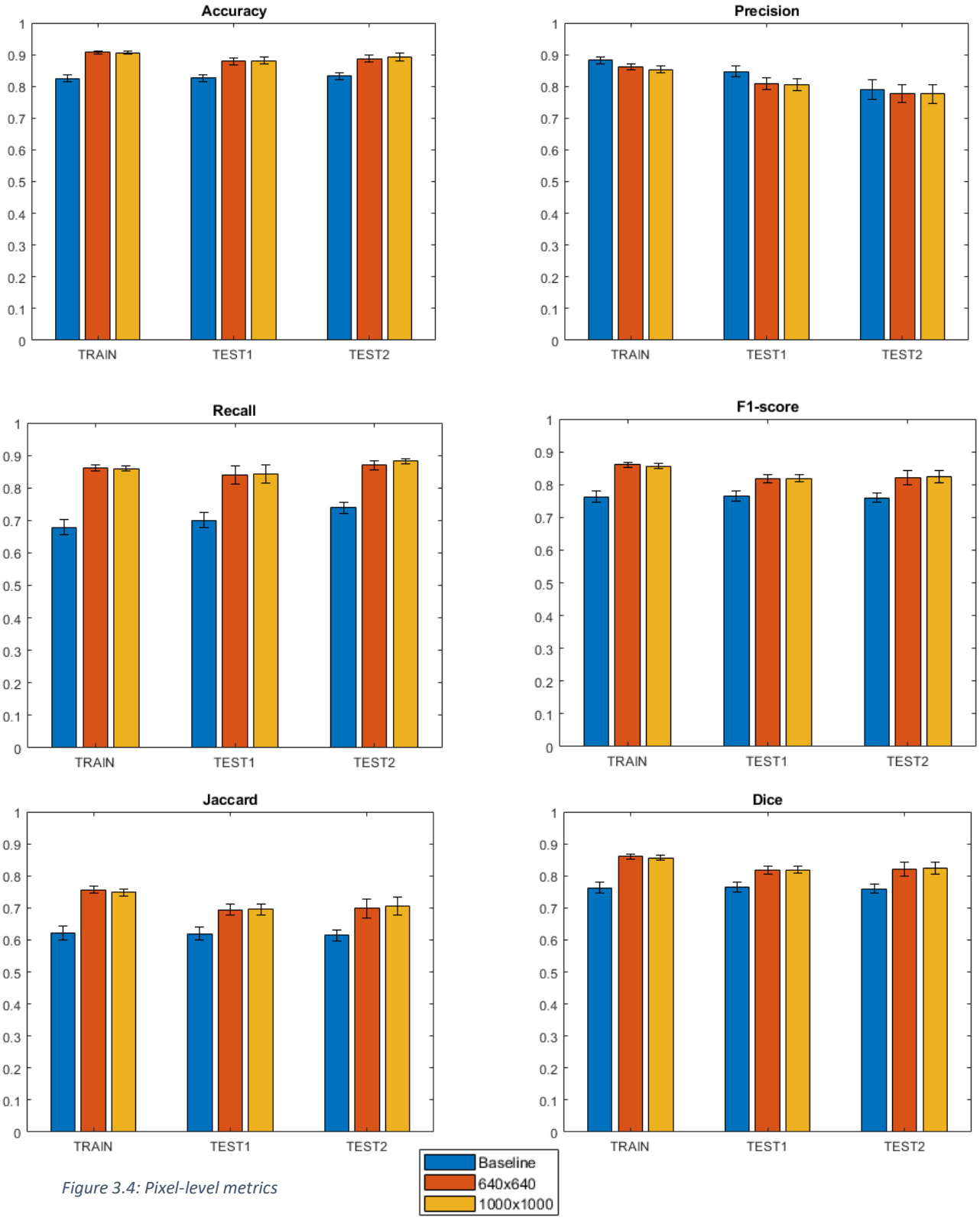


Figure 3.4: Pixel-level metrics

The bar graphs report the average value of the six pixel based indicators on all the three sets, along with the associated standard error. A consistent increase in performance can be observed for the accuracy (which actually already had a value higher than 0.8 on the baseline), recall, and F1-score: this confirms how the use of active contours can lead to a better convergence of the segmentation towards the nuclear borders, increasing the net number of correctly classified pixels. The metric with the lowest values is Jaccard and it shows a comparatively lower increase when compared to other indicators: this is not unexpected, as Jaccard is related to overall shape and the final overall shape of each nuclear segmentation ultimately depend on the one found by the segmentation network and the convergence of snakes cannot significantly alter it. The only metric that shows a decline, although reduced in magnitude, is precision: this is most probably due to some FP detections that are not correctly removed from softmax during contour initialization. Although they do not evolve if initialized on a prevalently red region due to the lack of energy terms, their final segmentation still consists in their initial circular shape and those pixels all contribute to the observed reduction of precision. Comparing the baseline average recall with baseline average precision, a general tendency of the segmentation network to under-segment nuclear regions when the raw segmentation is applied can be highlighted. For all pixel-level metrics no difference between the two models is observable. In addition, as opposed to the previous detection graphs, the behaviour among the three sets is more homogeneous.

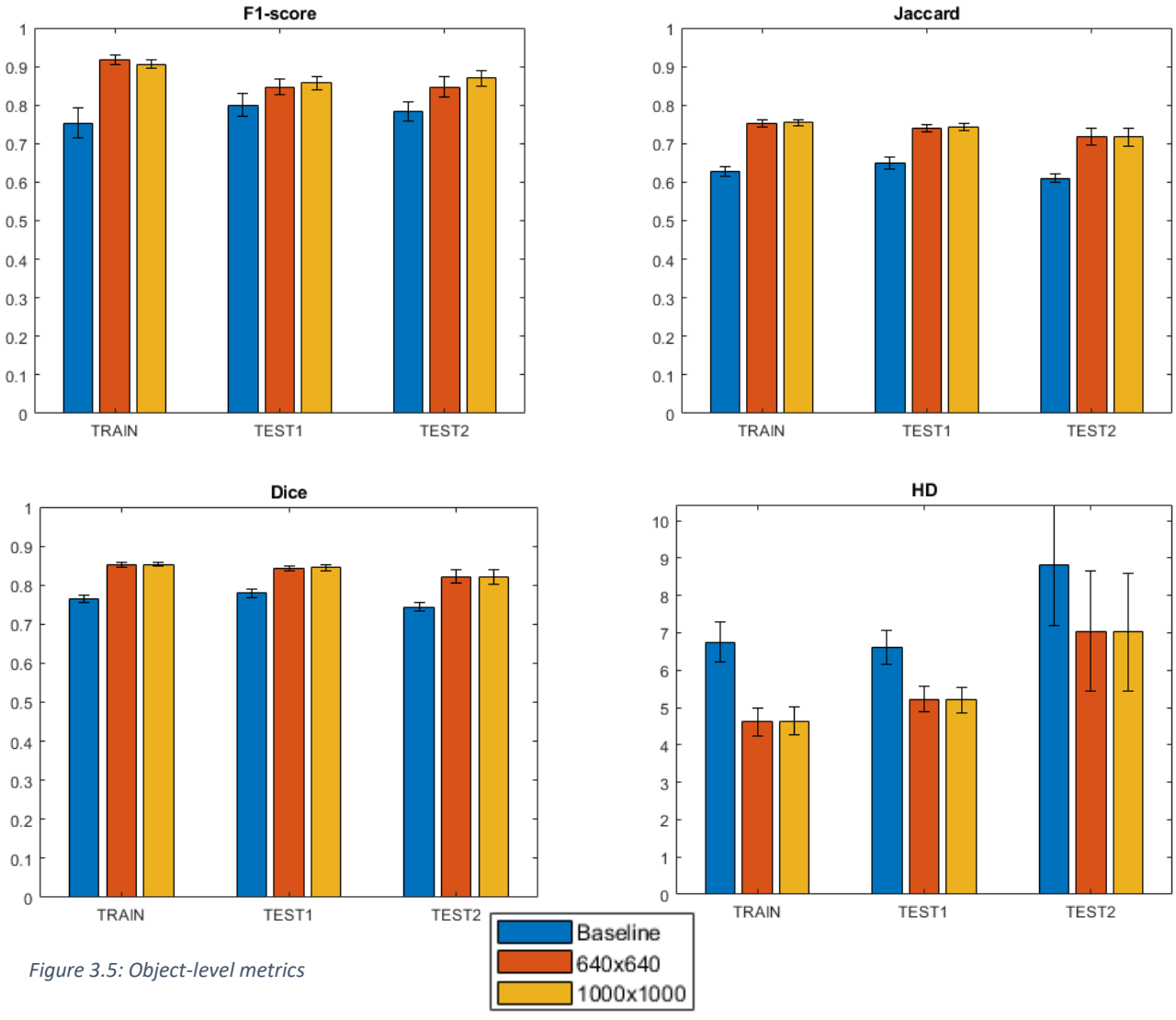


Figure 3.5: Object-level metrics

As far as the object based indicators are concerned, an in increment in performance is observed on all calculated metrics. Among the four, the most important to look at is probably F1-score, as the way it is calculated makes it a fully instance-based metrics, whereas all the other ones still inegrate some pixel-level information, despite being averaged on the single objects. It is hence the most useful to underline a second time the detection model performance. There, an average 7% to 15% increment is noticeable, with standard error values conserved or reduced compared to the baseline. This suggests that the detection network is indeed helping in the localization of a higher number of nuclei in the dataset: this effect has to be in the most part due to the now separate segmentation of previously fused instances, because the detection model cannot segment by itsef an identified nucleus, even when it's correctly placed, if the segmentation network missed it during softmax generation. Again, the effect of overfitting that affects the detection model can be here noted again with a similar, comparatively reduced increment of F1-score. DSC and Jaccard mainly bring redundant information to their pixel-level counterparts, with Jaccard again showing the lowest values in percentage term, for the same reasons previously stated. In addition, they show very consistent values across the dataset splits. HD metric shows an average decrement of around 3 pixels (the lower the better, in this case), with significantly higher standard error on the second test set, both for the baseline and the proposed results. This is probably a direct consequence of the intrinsic higher variability of this set, featuring nuclei with highly different sizes and shapes through its six images. This reduction in HD means that, on average, the maximal point-wise error between the final segmented instance and its ground truth is lower when the segmentation is obtained through the proposed active contours evolution.

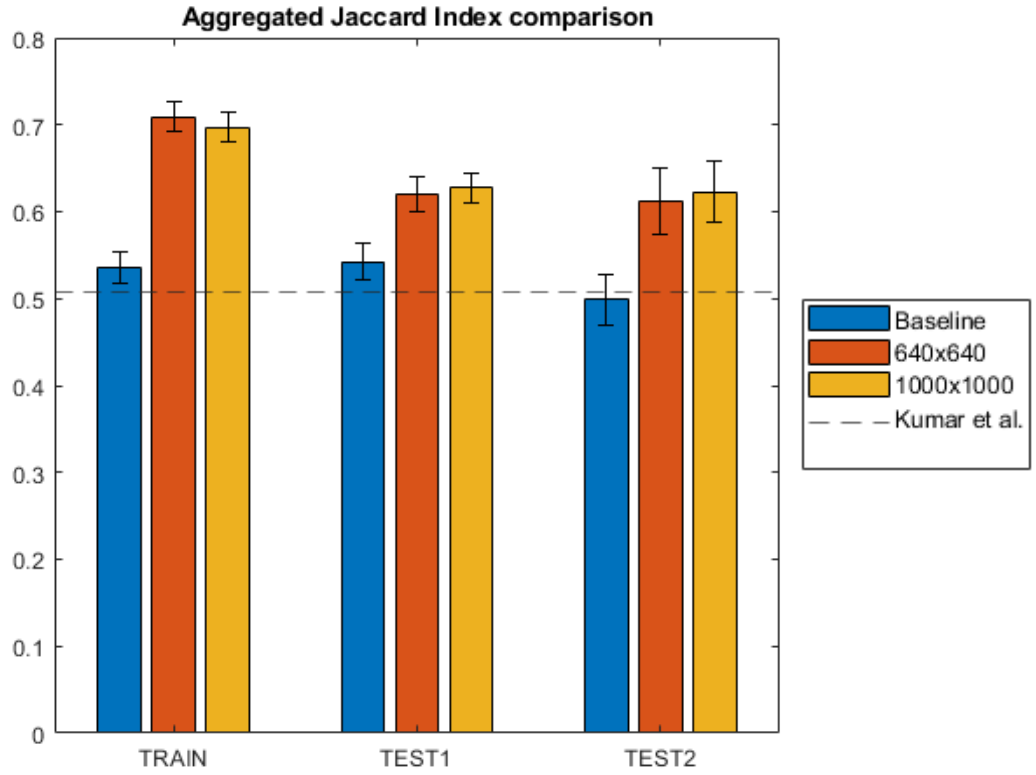


Figure 3.6: AJI

Finally, the results as computed by AJI are presented. The baseline values of AJI are in line or already marginally better than those reported by Kumar in his paper (dashed black line on the bar graph, computed on the two test sets). This proves that the softmax maps, that served as a starting point for this work, are already of good quality even when considered by themselves. The additional proposed framework steps grant an increase in AJI of about 0.16 for the training/validation set and about 0.10 on the two test sets, reaching an average value of 0.627. Once again, it can be observed that the detection network overfitting is carried over on the final segmentation, as the diffrence in performance between training and the two test sets is increased compared to the baseline. In spite of this, considering the strict nature of AJI indicator, which quickly stops increasing as even one of the four main errors listed in the introduction to paragraph 2.6.2 becomes prevalent over the others, the proposed framework still significantly improve the segmentation quality on the whole dataset. Surprisingly, the data split that receives the least amount of performance increase from the method is the first test set, whereas test set 2 starts from a lower baseline and reach comparable performance. Overall, the model trained on the full 1000x1000 images seems to feature marginally higher performances and to be less affected by overfitting, as its results are slightly more balanced through the training and test sets.

3.2.1 Statistical analysis of benefits

To further validate the proposed framework significancy, a brief statistical analysis was performed. This was done by applying a paired Student's t-test to the performance distributions prior and after the application of the pipeline. This is a common technique to test two given, assumed normal, distributions against a null hypothesis of equal means, within a predefined significance threshold level α (almost always set to 5% for standardisation and comparability purposes). In our specific case in exam, we test against the null hypothesis that the AJI performance calcualted after the framework application is part of the same dsitrbution of the baseline performance. In that case, the performance variations could not be fully credited to the proposed method, but they could simply be the result of intrinsic variability that lies in the distribution. The results for such a test are reported in the table below.

	p-value	
	640x640	1000x1000
TRAIN	0	0
TEST 1	0.023	0.011
TEST 2	0.052	0.03

The results hint at how the performance variations from the baseline can be considered as statistically significant in all cases except for the second test set when using the detector trained on 640x640 patches, where a p-value just above the significance level of 0.05 is obtained. This is probably due to the more variable results scored on the images composing the set (indeed, the standard error on most previosly commented bar graphs is higher for this set if compared to all the others). All in all, the t-test thus confirm the primary role of the introduced segmentation procedure in increasing the performance, at least on the employed dataset.

3.2.2 Potential maximum performance increase

To assess the peak potential benefits that could be brought by the proposed framework, in particular regarding the idea of integrating a second deep learning method in the form of an object detector in parallel to the pre-existing segmentation network, the performance in terms of AJI are recomputed after introducing a theoretically perfect object detector on top of the pipeline. This kind of evaluation also makes it possible to estimate the degree of feasible improvement that can be attained by acting on the object detection block of the pipeline. The results shown below are hence obtained by initializing the snakes directly on the bounding box centroid of each annotated region, which is exactly what a good detector is expected to do in this framework. The contours are then normally evolved using the same parameters and all the metrics are computed. Here, only AJI is reported as the most comprehensive indicator of performance.

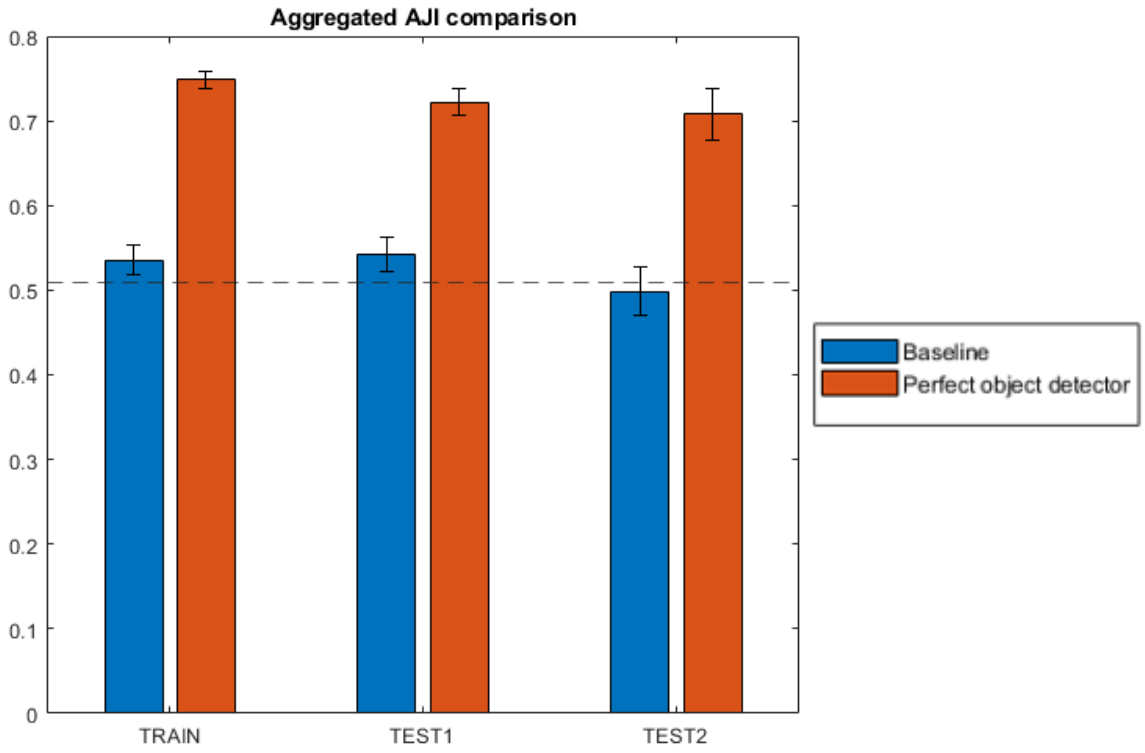


Figure 3.7: AJI with a perfect detector

The results show how there is still a good room for improvement as far as the object detector is concerned, reaching an average value around 0.7 AJI for the two test sets when using an ideal detection model. Considering that the values on the training set are already around that level for both trained detectors, the main front of further work in this context should be increasing the generalization capability of the model. It is important to keep in mind though that the results above are only asympotically reachable as they are based on a non-existing ideal model and to actually have a chance to push the results of the proposed framework to those performance values or beyond, simultaneous improvemenst would have to be introduced on the segmentation network that outputs the baseline softmax.

3.2.3 Algorithm steps visualization

With the intent of recapping the main blocks involved in the proposed framawork and better visualize the whole process, a step by step visual representation of the workflow will be inluded as the final part of this chapter. For a better understanding, a zoomed-in portion of one histological image from the dataset is used. The process starts from the color normalized version of the image and ends with the final comparison of the segmentation obtained from the application of the full pipeline versus the raw segentation. The intermediate processes of object detection, snake inizialization and contour convergence are also included. The steps below are numerated as follows:

1a) Ground truth annotations

1b) Color noramlized image

2a) Segmntation network output

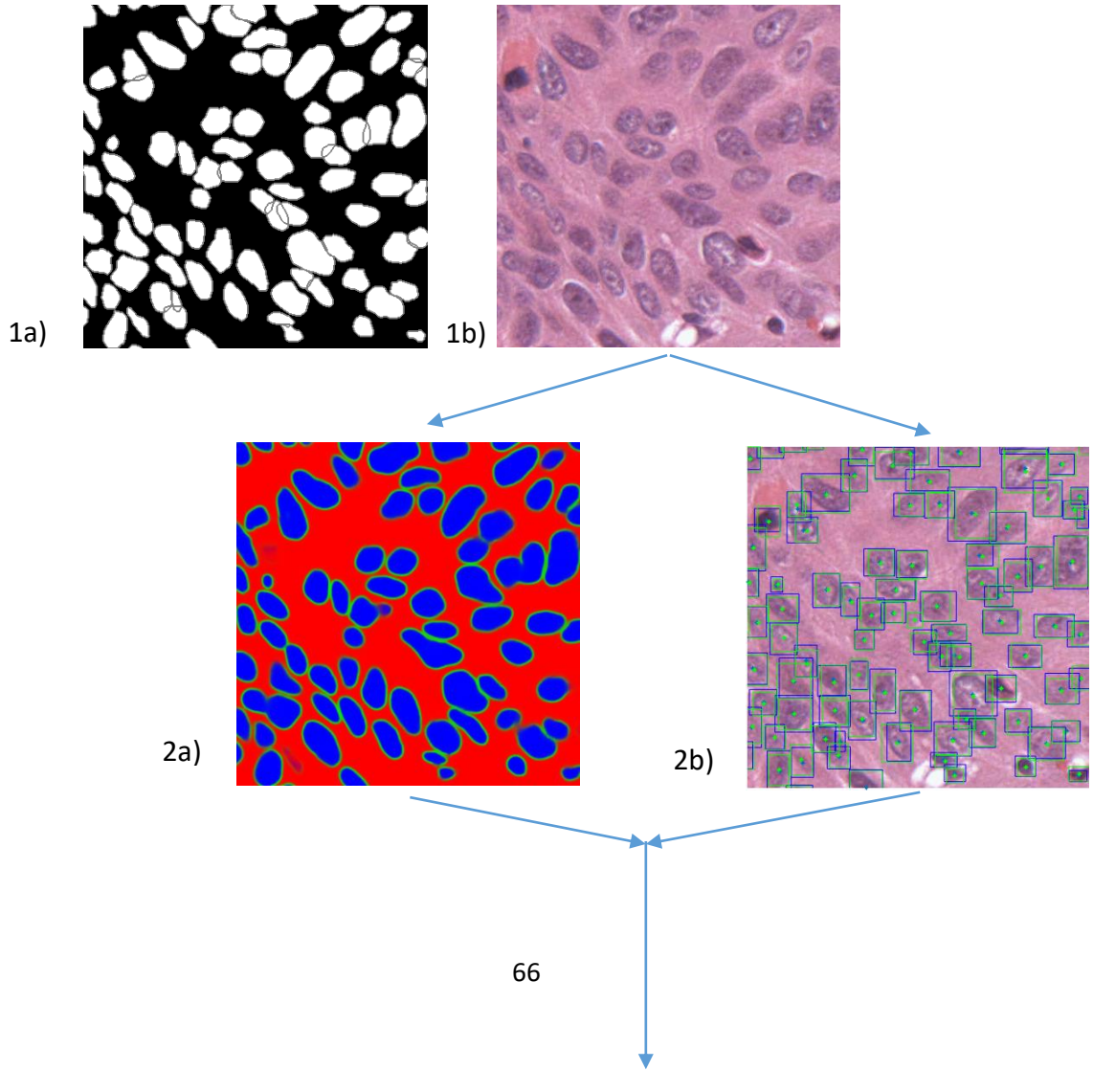
2b) Object detection network

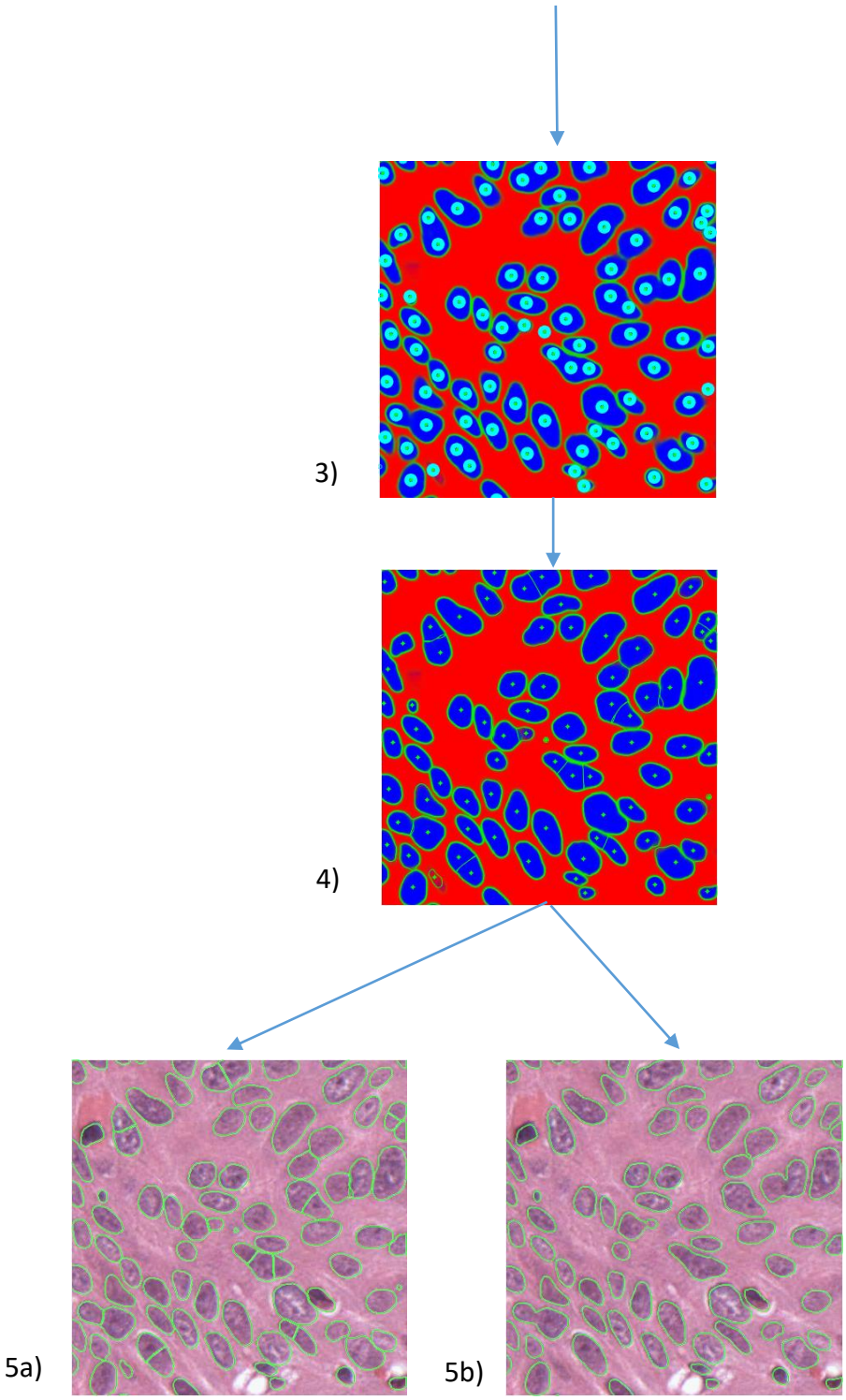
3) Active contours initialization

4) Contours evolution and convergence

5a) Proposed method segmentation

5b) Raw segmentation





The portion of image presents many close nuclei that are fused together in the raw segmentation, whereas they are correctly segmented by the proposed framework.

Conclusions and further work

4.1 Conclusions

This thesis project consisted in the proposal and development of a complete framework for the automatic segmentation of histological images. In particular, the focus was put on improving segmentation performance of crowded regions, such as those containing a high number of nuclei, where the typical CNN based deep learning algorithms tend to fuse together close or overlapping objects. This framework is hence based on the integration of deep learning techniques, consisting in one deep segmentation network and one object detection model, with a more traditional post-processing block based on parametric active contours. The introduction of the detection model is responsible of correctly localizing the presence of multiple close nuclei in the histological image, whilst the post-processing step performs the actual segmentation by evolving the contours on the softmax probability map. The proposed pipeline shows promising performance by scoring an average of 62.7% Aggregated Jaccard Index on the two test sets in use, with an increment of more than 10% on the segmentation baseline, obtained through a common softmax thresholding. The method thus proves to be beneficial in the improvement of pre-existing softmax segmentations. It does not reach the performance of 0.69 AJI of the top scoring participant of a competition based on the same dataset that was used on this thesis, but the results are not directly comparable as some of the training examples and all the testing images are different. Moreover, the main purpose of this work is not pushing state-of-the-art in nuclear segmentation tasks, as much as building a versatile framework to improve the output of traditional and established deep learning techniques. The pipeline is easily adapted to other medical imaging segmentation tasks, thanks to the high generality of the deep learning methods that were used. The additional computational time due to the object detection step is under 500 ms inference per image for the object detection step on a single Tesla K80 GPU (thanks to the use of one of the fastest one-stage architectures available), while the subsequent snakes' evolution step for segmentation takes 38.9 ± 5 seconds running on an Intel i3-9100F CPU. The best performances are obtained by giving the full 1000x1000 images of the dataset as input to the detection model: while this might seem counterintuitive at first, it is important to consider that such a resolution still represents a rather narrow FOV, when compared to the total size of a single WSI. As a consequence, the application of this framework at a higher scale on histopathological images would still require the adoption of a patch-based approach. All in all, the developed framework can offer and generalizable performance improvements when applied to any segmentation task featuring a high density of objects, without significantly slowing down the process. However, in spite of the highlighted potentiality, an eventual integration of the algorithm into a full digital pathology analysis tool requires further work on several fronts, ranging from performance improvements to computational optimizations.

4.2 Further work

In the future, several factors could be considered when trying to improve the proposed hybrid framework. These improvements should cover all the three main blocks that compose the segmentation pipeline:

1. Object detection
2. Softmax generation through deep CNNs
3. Active contours initialization and evolution

As far as object detection is concerned, the main improvements should point towards increasing the model generalization ability, to reduce the observed overfitting. To some extent, along with increasing the general performance, this could be addressed by a more in-depth tuning of the model hyperparameters used during training. Another viable option to explore is the implementation of a different and more sophisticated patch extraction mechanism during the pre-processing step. For example, the extraction could be denser around regions where a high number of nuclei are present, or it could in alternative simply include more examples that proved to pose a challenge for the model to learn. Eventually, other models outside of the YOLO family can be evaluated, in particular two-stage object detectors which should sport higher peak performance when properly optimized. At the current state, inference time of the model does represent only a small part of the total method runtime (provided that a GPU is used), so the additional computational burden of two-stage solutions may not be concerning.
Regarding the active contours step, their implementation has to be furtherly optimized to minimize computational complexity. As matter of fact, the current implementation makes up for the majority of the framework runtime (excluding the one-time training of the DL models and the evaluation of performance), and half of this time is used by the function computing the mutual interactions between snakes. This represent the biggest bottleneck in the current workflow and it must hence be the starting point for this code optimization process. The interaction term should also be tuned to allow for more natural segmentation when interaction between more than two snakes are involved.
Considering the nature of the proposed method, the individual optimization of only one aspect quickly leads to diminished performance returns. Indeed, a limited detection model causes the initialization of a wrong number of contours, effectively wasting the quality of the underlying softmax. In a similar way, the segmentation network poses a limit on the highest performance benefit that the framework can yield, as underlined in chapter 3.2.2, because the contours evolution is guided by the softmax itself. It follows that an effective optimization process has to consider the enhancement of the segmentation network in parallel to the object detection improvements hinted above.
Additional work may include the application of the current method to full WSIs and the experimental evaluation of the pipeline on different medical imaging domains. In the first case, some scalability problems are likely to arise without further modifications to the pipeline, due to the huge amount of data to segment. On the other hand, given the substantial generality of the framework, the extension to most domains should not carry particular challenges: it is in fact sufficient to retrain both the deep learning models with enough relevant data from the new domain, without the need to introduce particular modifications to downstream blocks.

Acknowledgements

Now that I'm eventually at the end of this journey, which gave me the possibility to grow up both personally and professionally, I would like to thank all the people with whom I shared a valuable part of my life and who helped me to reach this accomplishment. My first acknowledgements go to Prof. Filippo Molinari for introducing me to the contents of this thesis and for sparkling my interest on the topics of image processing during his teaching activity. A special thanks to my co-supervisor Ing. Massimo Salvi for all the support and for patiently guiding me through the writing of this work. Thanks to my sister and my closest friends that supported me outside of the academic world for their constant presence and helping hand during rough times. My biggest gratitude undoubtedly goes to my parents: their encouragement through the years has kept me motivated and I could not have undertaken this path without their unconditioned support and love.

Bibliography

[1] Subramanian, Muthukumar & Kumar, S N. (2017). A voyage on medical image segmentation algorithms. Biomedical Research. 2018.

[2] Yanase, Juri & Triantaphyllou, Evangelos. (2019). A Systematic Survey of Computer-Aided Diagnosis in Medicine: Past and Present Developments. Expert Systems with Applications. 138. 112821. 10.1016/j.eswa.2019.112821.

[3] Kim M, Yun J, Cho Y, et al. Deep Learning in Medical Imaging [published correction appears in Neurospine. 2020 Jun;17(2):471-472]. Neurospine. 2019;16(4):657-668. doi:10.14245/ns.1938396.198

[4] Salvi, Massimo & Acharya, U Rajendra & Molinari, Filippo & Meiburger, Kristen. (2020). The impact of pre- and post-image processing techniques on deep learning frameworks: A comprehensive review for digital pathology image analysis. Computers in Biology and Medicine. 128. 104129. 10.1016/j.compbiomed.2020.104129.

[5] Maximow, Alexander A.; Bloom, William (1957). A textbook of Histology (Seventh ed.). Philadelphia: W. B. Saunders Company.

[6] https://bitesizebio.com/13398/how-histology-slides-are-prepared/ (online)

[7] Mark D. Zarella, Douglas Bowman;, Famke Aeffner, Navid Farahani, Albert Xthona;, Syeda Fatima Absar, Anil Parwani, Marilyn Bui, Douglas J. Hartman; A Practical Guide to Whole Slide Imaging: A White Paper From the Digital Pathology Association. Arch Pathol Lab Med 1 February 2019; 143 (2): 222–234.

[8] Silverstein MJ, Recht A, Lagios MD, Bleiweiss IJ, Blumencranz PW, Gizienski T, et al. Special report: consensus conference III. Image-detected breast cancer: state-of-the-art diagnosis and treatment. J Am Coll Surg 2009;209:504–520.

[9] Bussolati G. Proper detection of the nuclear shape: ways and significance. Rom J Morphol Embryol. 2008;49(4):435-9. PMID: 19050790.

[10] Gurcan MN, Boucheron LE, Can A, Madabhushi A, Rajpoot NM, Yener B. Histopathological image analysis: a review. IEEE Rev Biomed Eng. 2009;2:147-171. doi:10.1109/RBME.2009.2034865

[11] Daisuke Komura, Shumpei Ishikawa, Machine Learning Methods for Histopathological Image Analysis, Computational and Structural Biotechnology Journal, Volume 16, 2018, Pages 34-42, ISSN 2001-0370.

[12] Gurcan MN, Boucheron LE, Can A, Madabhushi A, Rajpoot NM, Yener B. Histopathological image analysis: a review. IEEE Rev Biomed Eng. 2009;2:147-171. doi:10.1109/RBME.2009.2034865.

[13] Tizhoosh HR, Pantanowitz L. Artificial Intelligence and Digital Pathology: Challenges and Opportunities. J Pathol Inform. 2018;9:38. Published 2018 Nov 14. doi:10.4103/jpi.jpi_53_18

[14] X. Du, Y. Cai, S. Wang and L. Zhang, Overview of deep learning, 2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC), 2016, pp. 159-164, doi: 10.1109/YAC.2016.7804882.

[15] Wehle, Hans-Dieter. (2017). Machine Learning, Deep Learning, and AI: What's the Difference?.

[16] Christian Janiesch, Patrick Zschech, Kai Heinrich, Machine learning and deep learning, arXiv:2104.05314

[17] https://sitn.hms.harvard.edu/flash/2017/history-artificial-intelligence/ (online)

[18] Grossi, Enzo & Buscema, Massimo. (2008). Introduction to artificial neural networks. European journal of gastroenterology & hepatology. 19. 1046-54. 10.1097/MEG.0b013e3282f198a0.

[19] Johannes Lederer, Activation Functions in Artificial Neural Networks: A Systematic Overview, arXiv:2101.09957.

[20] Hecht-Nielsen, "Theory of the backpropagation neural network," International 1989 Joint Conference on Neural Networks, 1989, pp. 593-605 vol.1, doi: 10.1109/IJCNN.1989.118638.

[21] Arshak, Yousif. (2021). Optimization Algorithms in Neural Network: A Review.

[22] Keiron O'Shea, Ryan Nash, An Introduction to Convolutional Neural Networks, arXiv:1511.08458

[23] Ioffe, Sergey & Szegedy, Christian. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, arXiv:1502.03167.

[24] Kumar N, Verma R, Sharma S, Bhargava S, Vahadane A, Sethi A. A Dataset and a Technique for Generalized Nuclear Segmentation for Computational Pathology. IEEE Trans Med Imaging. 2017 Jul;36(7):1550-1560. doi: 10.1109/TMI.2017.2677499. Epub 2017 Mar 6. PMID: 28287963.

[25] Kumar, Neeraj & Verma, Ruchika & Sharma, Sanuj & Bhargava, Surabhi & Vahadane, Abhishek & Sethi, Amit. (2017). A Dataset and a Technique for Generalized Nuclear Segmentation for Computational Pathology. IEEE Transactions on Medical Imaging. 36. 1-1. 10.1109/TMI.2017.2677499.

[26] Rupprecht, C. et al. (2016), Deep Active Contours. ArXiv abs/1607.05074.

[27] Diego Marcos, Devis Tuia, Benjamin Kellenberger, Lisa Zhang, Min Bai, Renjie Liao, Raquel Urtasun, Learning deep structured active contours end-to-end, arXiv:1803.06329.

[28] Roy S, Kumar Jain A, Lal S, Kini J. A study about color normalization methods for histopathology images. Micron. 2018 Nov;114:42-61. doi: 10.1016/j.micron.2018.07.005. Epub 2018 Aug 1. PMID: 30096632.

[29] Thomas, Gabriel & Flores-Tapia, Daniel & Pistorius, Stephen. (2011). Histogram Specification: A Fast and Flexible Method to Process Digital Images. Instrumentation and Measurement, IEEE Transactions on. 60. 1565 - 1578. 10.1109/TIM.2010.2089110.

[30] Ruifrok AC, Johnston DA. Quantification of histochemical staining by color deconvolution. Anal Quant Cytol Histol. 2001 Aug;23(4):291-9. PMID: 11531144.

[31] Macenko, Marc & Niethammer, Marc & Marron, J. & Borland, David & Woosley, John & Guan, Xiaojun & Schmitt, Charles & Thomas, Nancy. (2009). A Method for Normalizing Histology Slides for Quantitative Analysis.. Proceedings - 2009 IEEE International Symposium on Biomedical Imaging: From Nano to Macro, ISBI 2009. 9. 1107-1110. 10.1109/ISBI.2009.5193250.

[32] Sinha, Rajat & Pandey, Ruchi & Pattnaik, Rohan. (2018). Deep Learning For Computer Vision Tasks: A review.

[33] Zhengxia Zou, Zhenwei Shi, Yuhong Guo, Jieping Ye, Object Detection in 20 Years: A Survey, arXiv:1905.05055

[34] L. Jiao et al., "A Survey of Deep Learning-Based Object Detection," in IEEE Access, vol. 7, pp. 128837-128868, 2019, doi: 10.1109/ACCESS.2019.2939201.

[35] Navaneeth Bodla, Bharat Singh, Rama Chellappa, Larry S. Davis, Improving Object Detection With One Line of Code (2017), arXiv:1704.04503.

[36] Solovyev, Roman & Wang, Weimin & Gabruseva, Tatiana. (2021). Weighted boxes fusion: Ensembling boxes from different object detection models. Image and Vision Computing. 107. 104117. 10.1016/j.imavis.2021.104117.

[37] Jan Hosang, Rodrigo Benenson, Bernt Schiele, Learning non-maximum suppression (2017), arXiv:1705.02950. NMS-Loss: Learning with Non-Maximum Suppression for Crowded Pedestrian Detection

[38] Zekun Luo, Zheng Fang, Sixiao Zheng, Yabiao Wang, Yanwei Fu, NMS-Loss: Learning with Non-Maximum Suppression for Crowded Pedestrian Detection (2021), arXiv:2106.02426.

[39] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi, You Only Look Once: Unified, Real-Time Object Detection (2016), arXiv:1506.02640.

[40] Min Lin, Qiang Chen, Shuicheng Yan, Network In Network (2014), arXiv:1312.4400.

[41] Joseph Redmon, Ali Farhadi, YOLO9000: Better, Faster, Stronger (2017), arXiv:1612.08242.

[42] R. Girshick, "Fast R-CNN," 2015 IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1440-1448, doi: 10.1109/ICCV.2015.169.

[43] Joseph Redmon, Ali Farhadi, YOLOv3: An Incremental Improvement (2018), arXiv:1804.02767.

[44] He, Kaiming & Zhang, Xiangyu & Ren, Shaoqing & Sun, Jian. (2016). Deep Residual Learning for Image Recognition. 770-778. 10.1109/CVPR.2016.90.

[45] Lin, Tsung-Yi & Dollar, Piotr & Girshick, Ross & He, Kaiming & Hariharan, Bharath & Belongie, Serge. (2017). Feature Pyramid Networks for Object Detection. 936-944. 10.1109/CVPR.2017.106.

[46] Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao, YOLOv4: Optimal Speed and Accuracy of Object Detection (2020), arXiv:2004.10934.

[47] Wang, Chien-Yao & Liao, Hong-yuan & Yeh, I-Hau & Wu, Yuen-Hua & Chen, Ping-Yang & Hsieh, Jun-Wei. (2019). CSPNet: A New Backbone that can Enhance Learning Capability of CNN.

[48] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, Jiaya Jia, Path Aggregation Network for Instance Segmentation (2018), arXiv:1803.01534.

[49] He K., Zhang X., Ren S., Sun J. (2014) Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. In: Fleet D., Pajdla T., Schiele B., Tuytelaars T. (eds) Computer Vision – ECCV 2014. ECCV 2014. Lecture Notes in Computer Science, vol 8691. Springer, Cham, 10.1007/978-3-319-10578-9_23.

[50] Diganta Misra, Mish: A Self Regularized Non-Monotonic Activation Function (2019), arXiv:1908.08681.

[51] https://github.com/ultralytics/yolov5 (online)

[52] Zheng, Zhaohui & Wang, Ping & Liu, Wei & Li, Jinze & Ye, Rongguang & Ren, Dongwei. (2020). Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression. Proceedings of the AAAI Conference on Artificial Intelligence. 34. 12993-13000. 10.1609/aaai.v34i07.6999.

[53] Kass, M., et al. (1988) Snakes: Active contour models. International Journal of Computer Vision, 1, 321-331 http://dx.doi.org/10.1007/BF00133570.

[54] Duan, Dingna & Zhang, H. & Qiu, C. & Xia, S.. (2015). A review of active contour model based image segmentation algorithms. Chinese Journal of Biomedical Engineering. 34. 445-454. 10.3969/j.issn.0258-8021.2015.04.009.

[55] S. S. Makhanov, "Active contours in medical image processing. Theory and applications," 2013 5th International Conference on Knowledge and Smart Technology (KST), 2013, pp. xviii-xxv, doi: 10.1109/KST.2013.6512772.

[56] Osher, S. and Sethian, J.A., Fronts Propagating with Curvature Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulations, J. Comput. Phys. 79, 12-49 (1988).

[57] Molinari F., Teaching slides from the course: “Medical image processing”.

[58] Cohen, Laurent D.. “On active contour models and balloons.” CVGIP Image Underst. 53 (1991): 211-218.