Berardo Mario Manzi

Segmentation and classification of breast cancer pathologies in histological images based on morphological patterns

Final Master's Project

directed by Prof. S. Romaní Also

DEIM

UNIVERSITAT
ROVIRA i VIRGILI

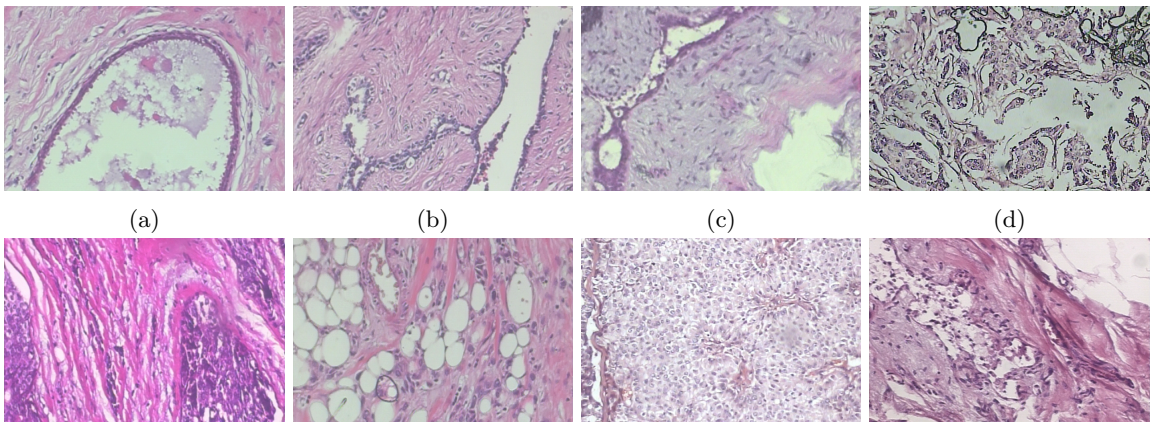
Tarragona 2018

Contents

Chapter 1

Introduction

Recommendations to prevent breast cancer for women in their 40s and older involves self-exams and mammography screening.[1] Eventual suspects of early- and late stage cancer types might require additional controls such as biopsies, to allow for histological studies. The process consists of extracting a small piece of tissue, stain it with Hematoxylin and Eosin (H&E staining),[2] slice it and acquire the images through the use of a camera connected to a microscope. Figure 1.1 shows typical examples of such images from the publicly available BreakHist database.[3]



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 1.1: Histological images of different types of breast cancer. First row are benign ((a) adenosis, (b) fibroadenoma, (c) phyllodes tumour and (d) tubular adenoma) while second row are malignant ((e) ductal carcinoma, (f) lobular carcinoma, (g) mucinous carcinoma, (h) papillary carcinoma). All images are taken under a 100x magnifying factor.

Expert pathologist may need to analyse hundreds of slices per patient, searching for areas where tumorous cells appear. Such an exhausting task, subject to misdiagnosis caused by fatigue, would benefit from Computer Aided Diagnose (CAD) systems as supporting tools for detecting and delineating the suspicious areas. Furthermore, CAD systems able to predict the disease type could reduce the workload for the expert, although human corroboration would still be required.

The design of a CAD system for breast cancer segmentation and classification is the final objective of this work. The task presents several challenges typical of Computer Vision (CV) and Machine Learning (ML) problems, plus others related to medical imaging. These issues will be outlined in the

forthcoming sections, together with the current state-of-art of the field.

1.1 Problem Setting

Image recognition, classification and segmentation are common tasks performed by the human eyes and visual cortex with relative ease, often without full awareness. However, transferring this capacity to computers has represented and still represents a major challenge in the field of Artificial Intelligence (AI).[4] Among the reasons of the difficulties is the different spaces the two visions systems act: human elaborate information in *semantic space*, while computers in *data space*.[5] The conversion from real perceived images to numerical information is usually obtained by representing colour as a set of numbers (e.g. Red, Green and Blue (RGB) space) at a given pixel position (i, j), which will subsequently undergo various transformations (normalisation, contrast enhancement) to make them available for high level feature extraction. Most of these modifications are problem specific, and approaches might be quite different according to the desired information.

The problem of dividing a set of images into a set of categories, or *classes*, is defined as **Image Classification**: figure 1.2 shows a set of images belonging to different classes and part of the CIFAR-10 dataset (https://www.cs.toronto.edu/~kriz/cifar.html).[6] These images show great intra-class variability (e.g. horses in different positions and of different colours) and inter-class similarities (e.g. white ships and white cars), but nevertheless a human being would distinguish them with ease. On the other hand, computer systems will need to quantify such differences in terms of pixel values, shapes, texture, and other visual related features, making the task all but trivial.

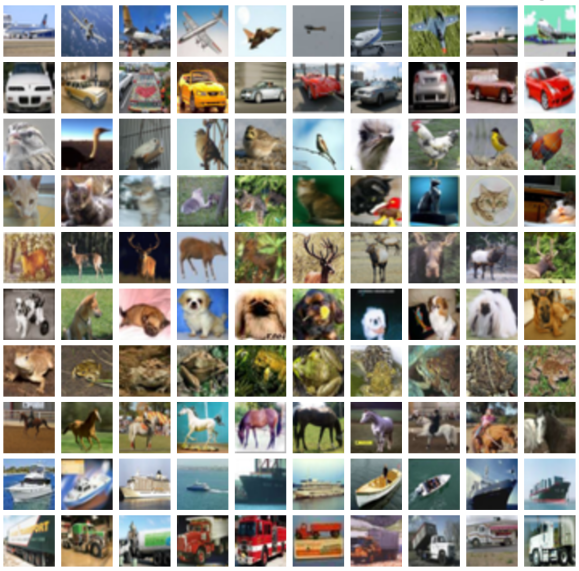


Figure 1.2: Examples of images used in Image Classification tasks. Each of the images belongs to one of the following 10 categories: plane, car, bird, cat, deer, dog, frog, horse, ship, truck. The image is part of the CIFAR-10 dataset (https://www.cs.toronto.edu/~kriz/cifar.html).

Analogous but perhaps more complex is **Image Segmentation**, i.e. distinguishing different categories composing a single image, as shown in figure 1.3. The increased complexity arises from the necessity of distinguishing categories on pixel by pixel basis, or at least on sub-patches of the whole image.

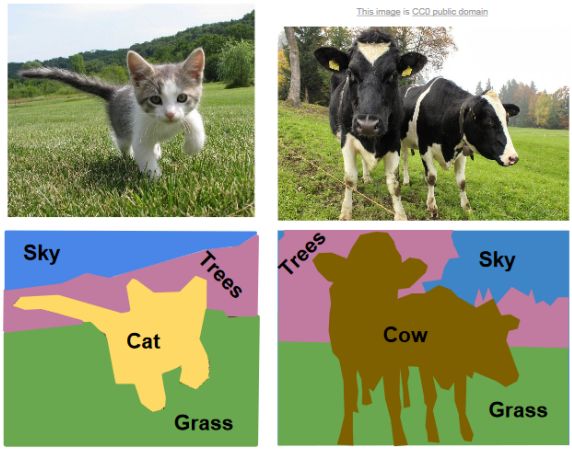
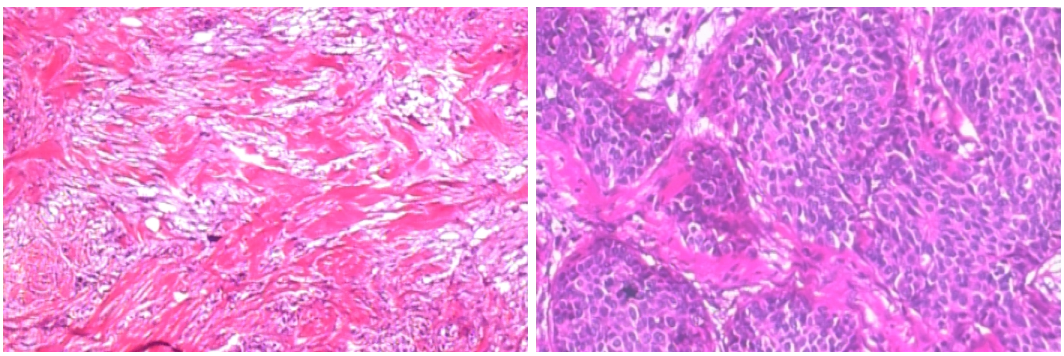


Figure 1.3: Example of Image Segmentation. Any algorithm designed for the task will be required to split the image into patches of pixels corresponding to each semantic entity. These images are part of the CS231n Stanford course (http://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture11.pdf).

The case of histological images treated in the present context is very similar, although some relevant differences are worth highlighting. First, images are annotated by expert pathologists, as proper medical training is necessary to distinguish healthy tissue from diseased one, and between different types of diseases. However, some degree of subjectivity is present in this analysis, and often disagreement between doctors might affect the accuracy of it. Furthermore, acquisition systems (i.e. cameras) are often not calibrated, introducing huge variability between images captured under different sampling conditions. For instance, the two slices in figure 1.4, despite displaying the same category (ductal carcinoma) are evidently taken under different lightning conditions, resulting in different colour shading for the different types of tissues (for instance, the pink colour is brighter on the left hand side picture than on the right). Pre-processing of the images can improve the recognition



(a) (b)

Figure 1.4: Examples of different acquisitions for the same class (ductal carcinoma) under different lightning conditions. Note the different contrast in *stroma* (pink colour) and epithelium (purple).

of these images, and is indeed a common step in the classification and segmentation tasks.

1.2 Related Work

Computer vision has been applied to medical images for more than 50 years, with hundreds of papers tackling the problem of cancer detection in histological images.[7] Several types of approaches have been exploited for the task of image analysis, recognition and classification, often relying on generic handcrafted features, i.e. manually engineered processes for detecting visual features like edges, colour, texture, morphology (shape, size) and so on. Typical algorithms for detecting such features include Local Binary Patterns, Co-occurrence Matrices, Wavelets, Active contours, Fuzzy clustering, Histogram of Gradients, etc.[4,8] These algorithms are commonly used in image processing, although some researchers suggested more domain related features, where the measurements stand for specific characteristics of tissue cells, like shape of nucleus, cell density, orientation and arrangement, gland angularity, and more.[9,10]

More recently, Deep Learning (DL) techniques have been applied to disease recognition in histological images,[11] where Convolutional Neural Networks (CNNs) are trained with thousands of sample images to assess which type of tumour corresponds to a given image or are utilised for segmentation of the tumorous areas inside the histological image.[12,13] Specific breast cancer problems tackled with CNNs are segmentation of nuclei, epithelium and tubules, detection of lymphocytes and mitotic cells, as well as classification of cancer genetic subtypes.[14]

DL methods have shown impressive prediction results, usually outperforming methods based on handcrafted features, whenever a great number of training examples are available. However, in the case of histopathological images, it is difficult to have access to such large ensembles of annotated samples, since it involves manual labelling of the images with the corresponding diseases provided by expert pathologists. Labelled images for segmentation are even more complicated to find, since every region of interest requires to be adequately marked up, a task more demanding than a attributing a single label per image.

On the other hand, CNNs are often criticised because the internal features tuned during the training are not understandable by medical personnel: CNNs can provide a diagnostic prediction but cannot tell the motivations, or facts that lead to and support said prediction (they suffer from lack of feature interpretability). Therefore, DL techniques with interpretable results, or so called *Explainable Artificial Intelligence* (XAI) approaches,[15] are desirable to make these CAD systems more appealing for real life applications. The present work aims to adhere to the XAI precepts, since our method is forced to learn some specific visual features that we think they are intrinsically related with the diseases to be detected and is described in the following chapters.

1.3 Project Objectives

The aim of this project is to provide a CAD system for breast cancer diagnoses from a set of histological images of a given patient. We have planned a two-stage system:

1. The first stage will segment a given histological image into areas of distinct types of cells, like epithelium, intra-lobular stroma, inter-lobular stroma and background. We hypothesised that a rather simple CNN would be able to provide an image segmentation like the ones shown in B. Ehteshami Bejnordi et al.,[13] where regions representing meaningful structures such as ducts and lobules are detected, but with less computational burden with respect to this reference.

2. The second stage will take the images labelled in the previous stage and try to classify them into the any of the following eight cancer types: adenosis, fibroadenoma, phyllodes tumour, tubular adenoma, ductal carcinoma, lobular carcinoma, mucinous carcinoma and papillary carcinoma (see figure 1.1) of which the first four are benign and the latter four malignant. The outcome of this stage will be compared to a more traditional end-to-end classification scheme, and we aim to prove that the two approaches will lead to similar accuracies.

The final goal of this two step system, apart from correct prediction of cancer classes, is to provide insight into the fundamental middle-level features distinguishing the cancer diseases, which we expect to be the morphology of biological structures made by the core types of cells (epithelium and stroma). Successful prediction on images with only morphological information (such as the outputs of stage one) would demonstrate which features are relevant to CNNs and which are not, giving some bits of explanation of the underlying mechanism of image recognition.

Noteworthy, our method is in line with the two-stage approach proposed in J. De Fauw et al.[16] where the authors also suggest a primary segmentation of the medical image (in their case, 3D Ocular coherence tomography scan) followed by a classification process to emit the final diagnosis. This approach has the advantage that it can highlight the unhealthy tissues onto the medical image to give clues to the therapists about the forecast disease. It is worth mentioning the we started the work on this idea before the reference was first published, which is a confirmation that our hypothesis is feasible and profitable.

Chapter 2

Artificial Neural Networks

Neural Networks have recently become mainstream, after the success of AlexNet, a deep convolutional network capable to outperform traditional image classification algorithms in the ImageNet challenge.[17] However, the theory of Artificial Neural Networks (ANN) has been researched for decades, dating back to the first model of human brain neurons.[18] Although this model has later been proven to be insufficient to describe the functions of biological neurons, it triggered a series of studies which lead to the modern concepts of ANNs.[19] Nevertheless, it was only until the advent of modern *Graphical Processing Units* (GPUs) that the predictive power of these models could be fully exploited.

The terminology utilised in the field of ANNs is reminiscent of concepts from neuroscience, despite the evident differences between ANNs and the human brain. For instance, the computational unit is named *neuron* but its "firing" is not time-dependent, contrarily to the mechanism in biological neurons. Further neuro-scientific concepts used when discussing ANNs are *receptive fields* and connections, or *synapses*.

This chapter will be devoted to the definition of the mathematics and the aforementioned concepts typically employed in ANNs, and more specifically in Convolutional Neural Networks (CNNs), the main topic of the remainder of this work. The forthcoming sections follow the approach described in J. Hertz, A.Krogh and R. G. Palmer,[19] and in the CS231n Stanford Course on CNNs for Visual Recognition (`http://cs231n.stanford.edu`).

2.1 Supervised Neural Networks

The ANNs used in our work belong to the classes of **supervised** neural networks, meaning that the outcome of the networks is required to match a certain set of labels or an interval of values, and it is trained until the target result is satisfactory. This is juxtaposed to **unsupervised** processes, where neural networks learn a certain set of features from the provided data, without previous hints on what these features are.

An example of a typical *feed-forward* architecture is depicted in figure 2.1. Each neuron in the *hidden* layer computes a linear combination of the inputs:

$$h_i^{(\alpha)} = \sum_{j=1}^{N_{\alpha-1}} w_{ij}^{(\alpha)} x_j + b_i^{(\alpha)}, \tag{2.1}$$

where i denotes the index of the neuron in layer α and the sum extends over the $N_{\alpha-1}$ neurons of the precedent layer. The actual output from neuron i is obtained by applying an *activation* function:

$$V_i^{(\alpha)} = g\left(h_i^{(\alpha)}\right). \tag{2.2}$$

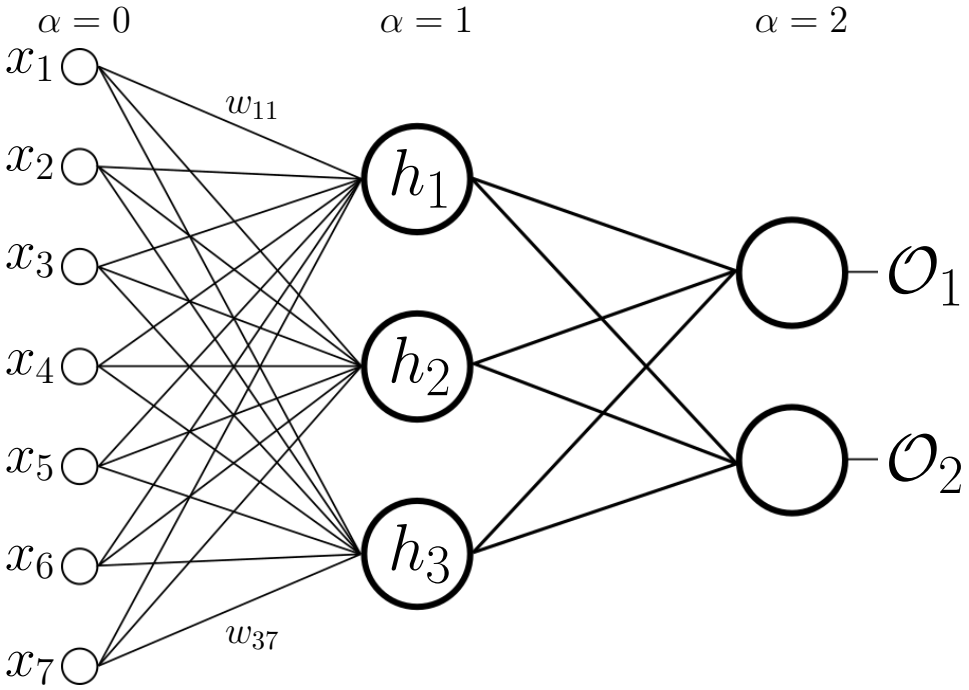


Figure 2.1: Illustration of a typical feed-forward Neural Network. The input values x_i are fully connected to the intermediate, or hidden, neurons h_i, i.e. each neuron is connected to all inputs. Analogously, the outputs $\mathcal{O}_i$ are connected to all hidden neurons. Each input from the previous layer is multiplied by the weight $w_{ij}^{(\alpha)}$.

The activation function $g(x)$ plays a fundamental role for the performance of the network, as it introduces non-linearity; in their original formulation W. S. McCulloch and W. Pitts[18] chose the Heaviside function:

$$\Theta(x) = \begin{cases} 1 & \text{if } x \leq 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.3}$$

However, common choices in more modern approaches are the sigmoid:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.4}$$

or, frequent in CNNs and utilised in this work, the *Rectified Linear Unit* (ReLU):

$$g(x) = \max(0, x) \tag{2.5}$$

which has been proven quite successful.[17]

Applying equations 2.1-2.2 for each neuron in each layer of the network leads to the final outputs, which, for the example illustrated in figure 2.1 yield:

$$\mathcal{O}_i = g\left(\sum_j w_{ij}^{(2)} V_j^{(1)} + b_i^2\right) = g\left(\sum_j w_{ij}^{(2)} g\left(\sum_{k=1} w_{jk}^{(1)} x_k + b_j^{(1)}\right) + b_i^2\right). \tag{2.6}$$

This equation represents a very general relation for prediction of the distribution of data, provided the weights $\{w_{ij}^{\alpha}\}$ and biases $\{b_i^{\alpha}\}$ are known. However, this is generally not the case, and the rather

complex dependence of $\mathcal{O}_i$ on these parameters requires an adequate way to adjust their values until a desired outcome is fitted. Hence, *Back-Propagation* is introduced.

Let's assume that we choose the values for weights and biases in a completely random fashion. The resulting output will certainly not match the expected outcome, but we are able to quantify this mismatch using a *Loss* function. For the sake of simplicity, we only introduce the *cross-entropy* loss used in our CNNs:

$$L_y = -\log\left(\frac{e^{f_y}}{\sum_j e^{f_j}}\right) \tag{2.7}$$

Here, we assume a **classification** problem, where we expect the network to output one of N_C discrete values, or classes. Thus, $\mathcal{O}_i$ will assign a score f_y to each of said classes, and the cross-entropy loss will depend on the ratio between the score of the expected class y and the sum of the scores of all classes.

Initially, L_y will be large, and minimising it, the mismatch between the expectation and the output will decrease. This can be achieved through classical numerical methods for finding minima of a function, such as *gradient descent*. In fact, computing the gradient:

$$\nabla L_y = \left(\frac{\partial L_y}{\partial w_{11}^{(1)}}, \ldots, \frac{\partial L_y}{\partial w_{mn}^{(\alpha)}}, \frac{\partial L_y}{\partial b_1^{(1)}}, \ldots, \frac{\partial L_y}{\partial b_m^{(\alpha)}}\right) \tag{2.8}$$

and updating the parameters:

$$w_{ij}^{\prime(\alpha)} = w_{ij}^{(\alpha)} + \eta \nabla L_y, \tag{2.9}$$

and analogously for the $b_i^{(\alpha)}$s will drive the function to its minimum. The parameter η is named the learning rate and has to be chosen with care, since small values will make the learning process excessively slow, while large values will deviate the updates too much at each iteration, possibly skipping over the target minimum.

The computation of the loss function and its gradient, followed by the modification of the weights, is applied recursively, feeding each time a different set of inputs $\{x_i\}$, called the *training* data. After the entire dataset is processed, the operations are repeated with the same data, over several periods, named *epochs*, until the accuracy of the predictions reaches values close to 1.

The concepts outlined thus far are sufficient to define a complete neural network architecture. However, there are several techniques necessary to optimise the learning process of ANNs, which we will introduce in the specific context of CNNs.

2.2 Convolutional Neural Networks

CNNs are a special case of neural networks, applied to problems of image classification and segmentation. Were we to process a $700 \times 460 = 322000$ pixels picture with the architecture of figure 2.1, we would require $3 \times 322000 = 966000$ weights at the first layer and the memory requirements would be very demanding. The solution is to use a different approach, inspired once more by biological neurons in the visual cortex. In fact, each of said neurons processes a subset of information received from the vision system (eye) and neighbouring neurons have some degree of overlap. Analogously, CNNs compute *convolutions*, an operation which consists of sliding an $m \times n$ matrix, or *filter*, F over the whole image I of size $M \times N$, and summing all products of each element of F with the element at the matching position of I. Figure 2.2 illustrates the effect of the operation. The actual entry in the final matrix, or *activation map*, is the result of applying an activation function, as in equation 2.2, to the output of the convolution operation, with eventual bias. Repeating the operation with L different filters yields L activation maps, equivalent to the N_α neurons present in *fully connected networks* (FCN), with the difference of having a matrix output instead of single value. It is evident that the

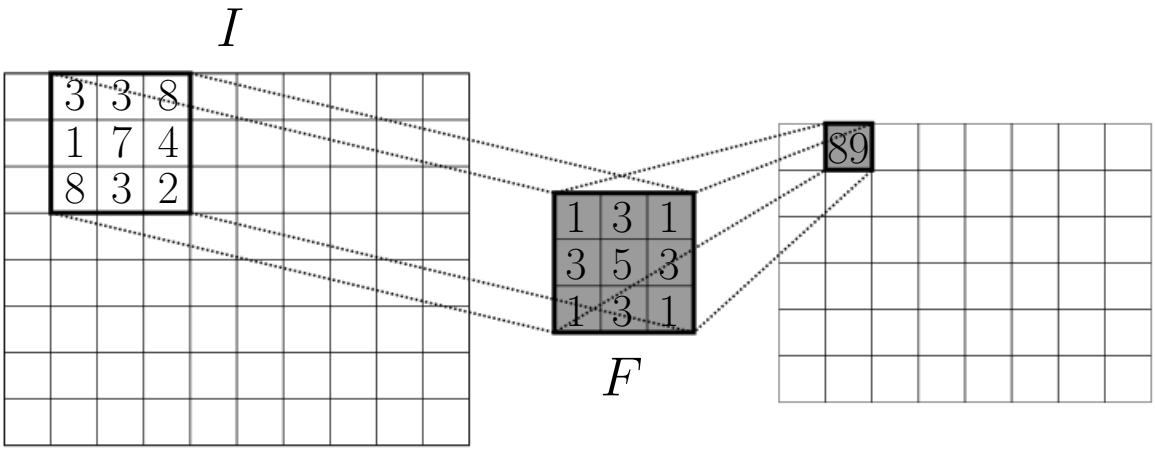


Figure 2.2: Illustrative explanation of the convolution operation. The highlighted 3×3 matrix F is slid over the 10×8 image I, each of the elements in F is multiplied by the corresponding element in I and the products are summed. Note that the image is an example of `'VALID'` padding, which produces an output matrix smaller than the original image.

main advantage of this approach is the decrease of trainable parameters. Consider, for example 16 filters of size 3×3, as in figure 2.2: the total amount of parameters, including 1 bias for each filter is $16 \times (3 \times 3 + 1) = 160$, about 3 orders of magnitudes smaller than with a FCN , where we need to define one connection per pixel. However, recalling that one of the final objectives is classification, we are still interested in reducing each image to a single label. Therefore, following several CNN layers, it is common practice to apply one or more fully connected (FC) layers, to obtain the single valued per class output. If no further considerations are made, we will be facing the same issues of huge parameter space, just shifted by some convolutional layers. Thus, *downsampling* is necessary.

Let's introduce this concept with an example. Figure 2.3 shows a CNN with 2 convolutional layers and 2 fully connected layers. The original image has dimensions $460 \times 700 \times d$, with $d = 3$ for a Red, Green, Blue (RGB) colour image, or $d = 1$ for a grayscale image. Then, we apply the convolution operation, using, for instance, 16 filters of dimensions 5×5. The output will be a set of 16 activation maps or, equivalently, a $460 \times 700 \times 16$ activation map, assuming `'SAME'` or *zero-padding*, i.e. adding rows and columns of zeros around the original image to get the desired output size. In fact, without padding or, equivalently, `'VALID'` padding, the first valid placement for the filter to overlap in its entirety the image (see figure 2.4) is at pixel $(2, 2)$ (origin at the top left pixel $(0, 0)$). Upon addition of a bias and an activation function, we proceed with down-sampling using *max pooling*. In practice, given a 2x2 window of pixels, only the greatest value is preserved, reducing effectively the output by a factor of two. Alternatively, it is possible to use a different *stride* while convolving, i.e. sliding the filters in steps of two, or even utilise a combination of padding, striding and pooling. Anyhow, at each step (either convolution or pool layer), the output width will be given by[1]:

$$w' = \frac{w - f + 2p}{s} + 1 \tag{2.10}$$

where w is the original width, f the filter size (e.g. 5), p is the amount of zeros added and s is the stride step. An analogous relation is valid for the height h of the image. In our case, $w = 700$, $f = 5$, $p = 2$ and $s = 1$ for the convolution layer, yielding $w' = 700$, while $f = 2$, $p = 0$ and $s = 2$ for the max pooling layer, resulting in a 350 pixel wide activation map.

[1] http://cs231n.github.io/convolutional-networks

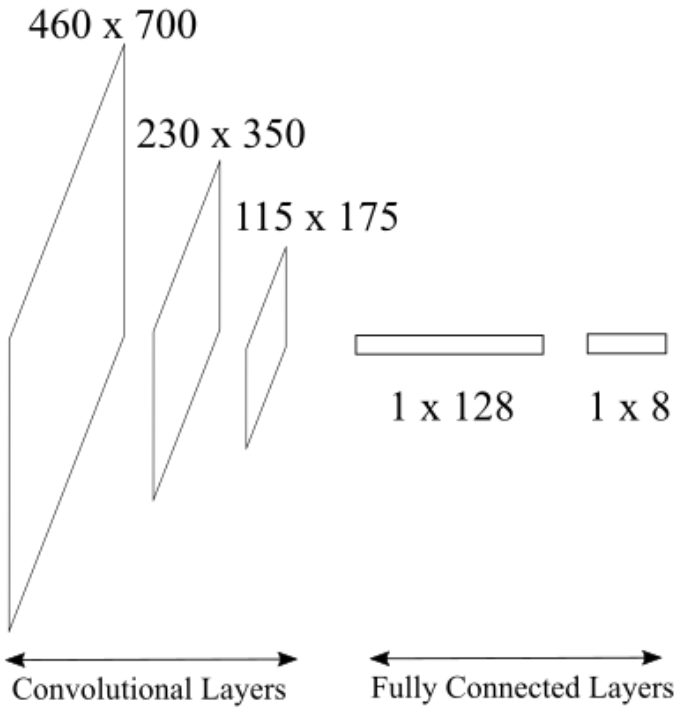


Figure 2.3: Example of CNN architecture. The initial image of width $w = 700$ pixels, height $h = 460$ pixels and depth $d = 3$ (1 for each RGB channel), is reduced to a 230×350 matrix upon convolution and downsampling. Another convolutional layer reduces the dimensions further, and the final 115×175 matrix is fed into the first fully connected layer, which outputs a scalar value for each neuron in the layer. The number of outputs of the last FC layer corresponds to the number of classes into which to classify.

Similarly, we apply convolution plus max pooling at each subsequent layer, until the first fully connected layer is reached. At this point, the matrix is generally reshaped to a single vector of values, and the reminder of the network behaves as a classical FCN. The last layer must be formed by an amount of neurons corresponding to the desired number of classes.

Note that the fully connected layers can also be considered convolutional layers, with the filter size matching the input size. This consideration has been exploited for image segmentation,[20] a task in which we are not interested in merely classifying an image as a whole, but rather identifying which region of the image contains said class. We will delve deeper into segmentation when discussing our objectives.

At this stage, it is convenient to explain some further concepts related to neural networks in general. Equation 2.9 describes the core of *training* a neural network, by updating the parameters according to the direction of the gradient of the loss function. Obviously, if the weights are all initialised to the same values, the update will be the same for all of them. Therefore, it is very common to draw the values from random distributions, either uniform or normal. Similarly, the biases are also randomised, and a similar updating rule is used.

Another issue related to expression 2.9 is the instability; in fact, the gradient might point in different directions in the parameters space, and updating the weights frequently will make the learning process unstable. Thus, it is common to compute the gradient for a set, or *batch*, of inputs (images) instead of a single one, and apply the update only once per batch. While this approach removes

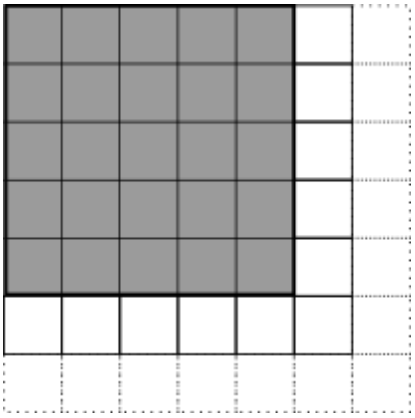


Figure 2.4: First valid position for a 5×5 filter without zero-padding.

some instability, the training process uses to be quite slow and, to fasten it, different optimisers are employed. For instance, adding a *momentum* term to equation 2.9 we get:

$$\Delta w'_{ij} = \eta \nabla L + \mu \Delta w_{ij} \tag{2.11}$$

will speed up the convergence. Indeed, the additional term takes into account the previous update of the weights, adding it to the update in the current epoch, acting, thus, as a sort of inertia term. Note also that we dropped the layer superscripts and classes subscripts for the sake of simplicity.

More advanced optimisers have been developed. An example is the Adam optimiser,[21] utilised in our work:

$$\Delta w_{ij}(t+1) = \eta(w_{ij}(t)) \nabla L, \tag{2.12}$$

where the learning rate η is now dependent on the updates of the steps precedent to $t+1$:

$$\eta(w_{ij}(t)) = \frac{\eta_0}{\sqrt{G_{t,ii} + \varepsilon}}. \tag{2.13}$$

Here, we refer to the constant learning rate of equation 2.9 as η_0 $G_{t,ii}$ is the diagonal matrix containing the squared sum of the gradients up to timestep t and $\varepsilon \ll 1$ is a common value introduced to avoid division by zero.

Nevertheless, these considerations are often not enough: further "tricks" consist in adding a regularization term to the loss function L:

$$L' = L + \lambda \sum_{i,j} w_{ij}^2 \tag{2.14}$$

with the second term reducing the absolute values of the weights. Finally, the input data is generally normalised, in a manner to equalise the importance of different features in the learning phase. For images, normalising can occur either by rescaling the colour values to the $[-1, 1]$ interval or by dividing the value by their standard deviation after zero centering, i.e. subtracting the mean of all pixels in a batch of images. The best working approach is problem specific.

Given a neural network which *learns* to classify or segment images from a training set, we will have to test if the process is a mere fitting of provided data, or the ANN actually has some predictive power. For this purpose, further images, belonging to the given classes, are fed to the trained network, which is supposed to identify correctly the class for each of them. If this is not occurring, the network has most likely *over-fitted* the training data, and is not considered to be properly trained. Thus, the

predictive power of an ANN is based on the accuracy of the classification of test images. Over-fitting can be fought using any combination of the following techniques:

Larger Training sets More training data most likely increases the variability of the patterns, making it harder for the network to specialise for some features absent in the test samples.

Data Augmentation Also useful for different reasons, data augmentation consists in rotating, scaling and transforming in different ways the original images, increasing the difficulty to learn specific patterns present only in the training samples.

Smaller Network Reducing the number of layers and/or the number of neurons per layer will weaken the learning power, reducing the over-specialisation of the network.

Dropout Excluding a certain amount of neurons with a given probability from a learning step diminishes the learning power.

Obviously, trying to fight over-fitting may result in under-fitting, making the network weak in both training and testing phase. The balanced choice will generally be determined by a trial and error approach.

The concepts outlined thus far are quite general, although this is far from being a comprehensive review of all possible algorithms and methods. Anyhow, the next sections will describe more in details the two approaches we employed introducing the basis to understand the results in the forthcoming chapters. A schematic representation of the two methods is shown in figure 2.5.

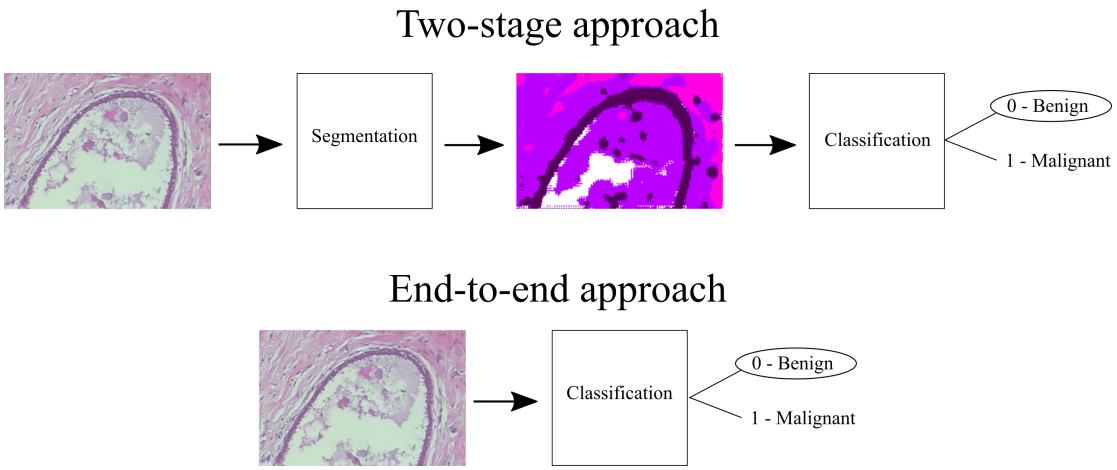


Figure 2.5: Schematic representation of the two-stage approach introduced in our work and a traditional end-to-end classification approach. The former splits the visual feature detection process into segmentation and classification, while the latter classifies without providing intermediate information, i. e. acting as a black box.

2.3 First approach: End-to-end Classification

The traditional approach to image classification consists in defining a neural network on the model of the CNNs described in the previous section, take a set of images, split it into training and testing subsets, and training the network. Although it is possible to recover some intermediate information like the representation of the filters, the overall process behaves as a sort of *black box*, which reduces

the appeal for non machine learning experts such as medical doctors, and hinders its applicability. However, to compare the performance with our two step segmentation plus classification method, we prepared two types of end-to-end classifiers: one capable of identifying benign from malignant breast tumours (2-class CNN), and one capable of classifying histological images into the eight categories: adenosis, fibroadenoma, phyllodes tumour, tubular adenoma (benign), ductal carcinoma, lobular carcinoma, mucinous carcinoma and papillary carcinoma (malignant). The latter defines an 8-class classifier which, as we will see, would require a larger data set than the BreakHis database[2].

The source code is implemented in the Python programming language[3], making use of the framework for machine learning called Tensorflow[4]. We implement a series of architectures, listed in chapter 4, following the scheme described above: 2 to 4 convolutional layers, followed by 2 fully connected layers. We demonstrate that greater number of layers do not improve the performance, and that we deem unnecessary to develop models like AlexNet,[17] with 8 layers, or even larger.

The networks we use employ `'VALID'` padding, to avoid artifacts related to pixels not carrying any information, and both approaches with max pooling and without have been systematically tested. Finally, the images are normalised using the standard deviation method explained in section 2.2.

2.4 Second approach: Segmentation method

Our main objective, as mentioned in the introduction, is to define a two-step segmentation plus classification method, which will result in an image classification algorithm with some insight into the motivations of a given class. For this purpose, we have to understand what are the important visual features that an expert pathologist would inspect and evaluate to diagnose a certain type of cancer.

The female breast is mainly composed of adipose tissue, ducts and lobes. A lobe is constituted of smaller structures called lobules, which produce milk in nursing women. Milk is then transported towards the nipple by the ducts. Lobules and ducts are the regions were cancer is most likely to occur.[22] Figure 2.6 shows an example of diseased breast tissue, at 100x microscope magnification. It is possible to observe the main components of the breast just mentioned:

- The duct in the left half of the image is shown in white, since the section of the tissue due to the cut leaves empty space, from here on denoted as **background**. Some traces of neighbouring tissue is present in the middle, most likely introduced from cutting during the preparation of the microscopic slide.
- Surrounding the duct, we recognise epithelial cells, having structural properties.
- Outside of the epithelial tissue, but still inside the lobules, the image displays connective tissue, interleaved with some empty space and some isolated epithelial cell. This part will be denoted as **epi-stroma**, being it a combination of epithelial cells and connective tissue (intra-lobular stroma).
- The connective tissue surrounding lobules is called **inter-lobular stroma** or simply stroma, and assumes the pinkish colour due to staining process.

The shape of these four structures, and their abundance, help pathologist to recognise the presence of cancer and its type. In fact, comparing figure 2.6 to figure 1.4, it is immediately evident that, in the latter, ducts are almost completely absent, since they have been filled by cancerous epithelial cells.

At this point, we like to emphasise the importance of choosing an adequate scale for classification and segmentation. Figure 2.7 shows histological slides from the same patient of figure 2.6, at 40x,

[2] https://web.inf.ufpr.br/vri/databases/breast-cancer-histopathological-database-breakhis
[3] https://www.python.org
[4] https://www.tensorflow.org

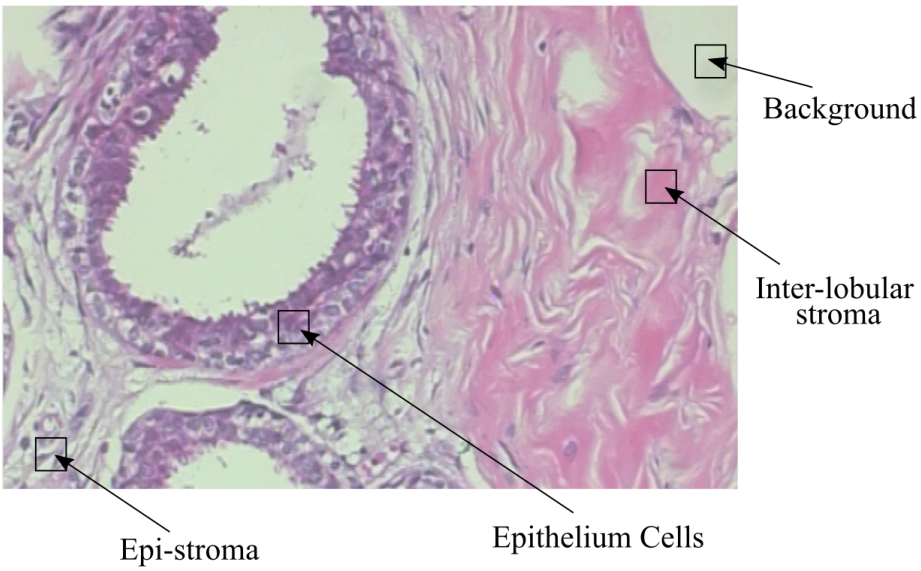
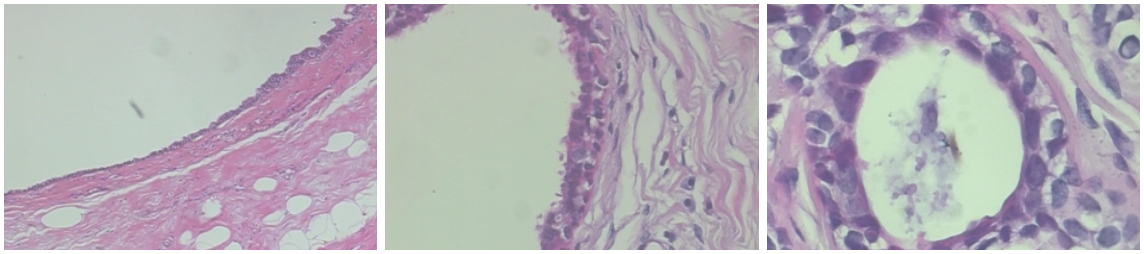


Figure 2.6: Histological image of breast tissue affected by benign adenosis tumour, at 100x magnification. Stroma represents non structural components, such as connective tissue and blood vessels. We distinguish two classes containing stroma: the *per se* component outside the lobules, named *inter-lobular stroma* and the component inside the lobules, often mixed with epithelial cells and thus unified in the epi-stroma class. Epithelium cells surround ducts, and the empty space in each cut defines the background.

200x and 400x magnification factor. We deemed the first scale as not magnified enough, because the epithelial cells appear excessively small. On the other hand, 400x magnification delivers too detailed information for single cells. To discern between 100x and 200x magnification we had to define the *receptive field* of our network. We decided to down-sample the original images by a factor of 4 on each dimension (height and width), which implies that each pixel in the CNN output image, will contain information previously stored in a 4×4 region of the original image. The actual receptive field will depend on the size of the filters, the stride and the pooling. Figure 2.8 shows an example of down-sampling using two layers, each applying a 4×4 filter and stride 2. The receptive field of a single output pixel is a 10×10 region of the input image. The idea behind the choice of 100x magnification factor is the request to have sufficient information from the surrounding of an epithelial cell, as well as enough information about the shape of each cell inside the receptive filed of each position of the output segmentation.

Having defined the set of images on which to deploy our network, the next step is to drive it to learn the aforementioned classes. For this purpose, we have down-sampled some of the BreakHis images and labelled the resulting pixels using five colours: white, pink, purple, violet and black, with RGB values $(255, 255, 255)$, $(185, 0, 255)$, $(85, 0, 93)$ and $(0, 0, 0)$, respectively. The black colour is introduced as an unknown class, meaning that pixels corresponding to this colour have to be excluded from training since it is not clear to which class they belong. The labelled version of figure 2.6, or *ground truth*, is shown in figure 2.9.

The size of the original images in the BreakHis database is 460×700 pixels, while our labelled pictures have dimensions of 115×175 pixels. The design of the network architecture must match the respective dimensions of input and output, independently on the amount of layers we are using. It is worth to notice that, contrarily to the classification case, the output of segmentation is an activation



(a) (b) (c)

Figure 2.7: Examples of breast tissue histological images at (a) 40x, (b) 200x and (c) 400x. Images taken from the same patient.

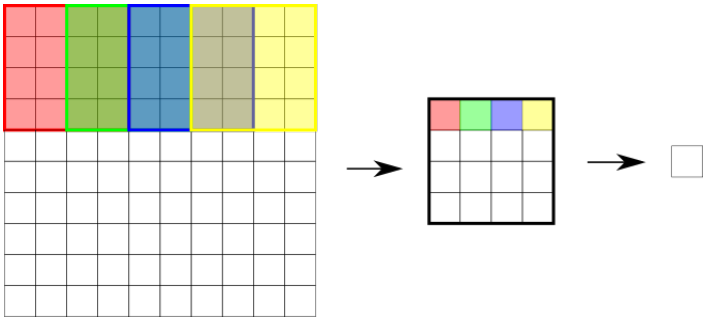


Figure 2.8: Illustrative explanation of receptive field, for a CNN of two layers, with size of filters 4×4 and stride 2. Each pixel of the output of the second layer is a convolution of a 4×4 region of the output of the first layer, and thus of a 10×10 region of the input image.

map, not a single value, given that it is the pixels we are classifying, not the image as a whole. Therefore, we will not be using any fully connected layers for the segmentation problem.

For the actual design we attempted three different approaches: CNN layers without and with max pooling, and CNN layers with a deconvolution layer, which perform up-sampling. This latter method has been inspired by the work of J. Long, E. Shelhamer and T. Darrell,[20] where the authors introduce the concept of *Fully Convolutional Network*. However, our approach differs from their work for the absence of pre-training: we start from a randomly initialised network and train it from scratch, with or without deconvolution.

Resuming, the procedure for Segmentation follows these steps:

1. Normalisation of the original images using their mean pixel value and the standard deviation.

2. Training of a L layered network, using convolutional layers, max pooling, ReLU activation function, Adam Optimiser and, optionally, a deconvolution layer. The 460×700 input images are down-sampled to 115×175 through the process, and the CNN prediction compared to hand labelled ground truth.

3. Testing of the network using further original and labelled images, and comparing the resulting accuracies of testing and training.

4. Improving the network by reducing over-fitting, using drop-out and data augmentation (rotation).

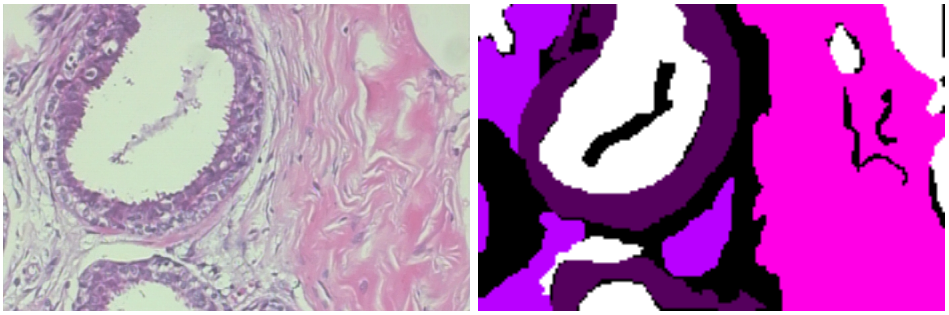


Figure 2.9: Example of labelling of a BreakHis image. The labelling is hand-made.

The outcome of the experiments are illustrated in chapter 3. Once trained, the network can be used to produce labelled images, with the accuracy obtained by segmentation. These images are then fed into the second stage.

Chapter 3

Segmentation Experiments

The results of the experiments concerning semantic segmentation of breast cancer histological images are presented in this chapter. At first, we will overview the dataset used for training and testing, discussing possible issues associated with the images which could hinder optimal performance. Then, we present the different architectures employed, and their capabilities of prediction.

Finding the "optimal" network architecture requires a lot of trial and error, as is well known in the ML community. However, every experiment performed needs to give some insight into how to change the parameters of the system, since blindly modifying them is unlikely to lead to a working solution. Therefore, we describe our observations for each of the experiments conducted, and explain what is the rationale behind the choice of parameters.

Finally, we conclude this chapter by testing our approach on a different set of images,[14] proving that our methodology is correct.

3.1 Description of Images and Ground Truth

The training and test dataset used for segmentation is composed of 7 images and corresponding labels, as reported in figures 3.1-3.2. Although such a small number of images appears to be insufficient, it is important to recall that each label in the ground truth corresponds to one of the four classes. Therefore, a ground truth image of 115×175 pixels corresponds to 20125 labels, with

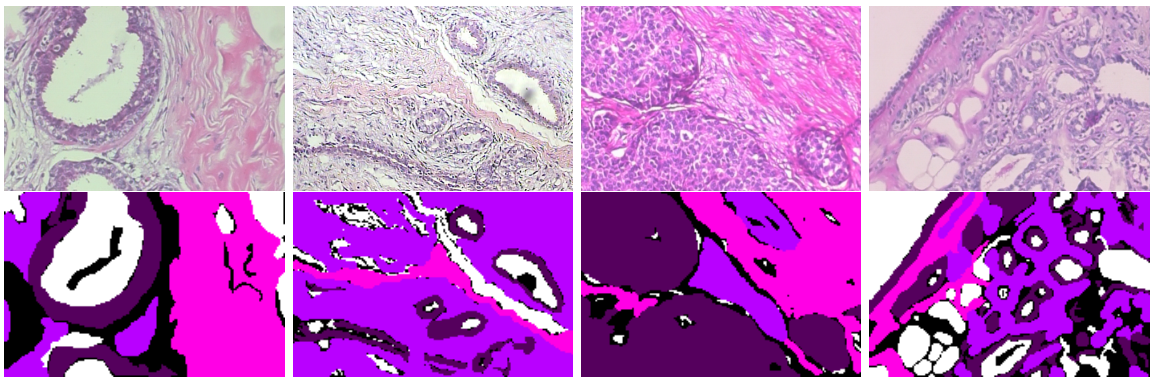


Figure 3.1: Histological images (top row) and corresponding ground truth images (bottom row) used for training.

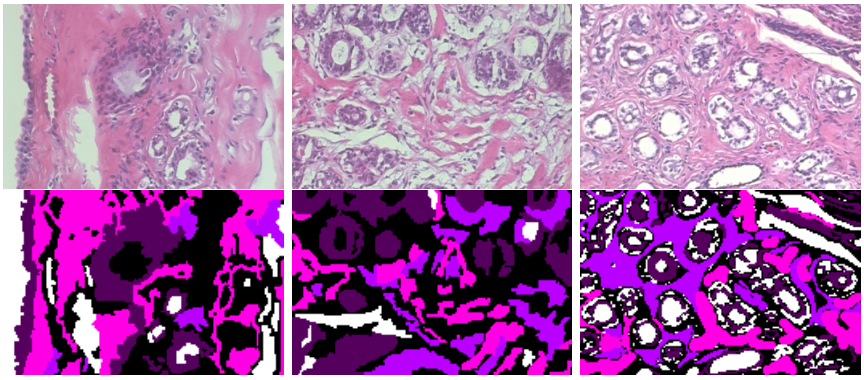


Figure 3.2: Histological images (top row) and corresponding ground truth images (bottom row) used for testing.

4 images resulting in 80500 samples. Obviously, in this total amount, we included the black labelled pixels, which, as explained in section 2.4, are not used for training, reducing the significant samples by about 10-15%; nevertheless, the total number of labels is large, and, as will be shown, sufficient for our segmentation problem.

The images were employed in two different forms during the training phase: as a whole, 115×175 image, or split into patches of varying sizes. The advantages of the latter approach is represented by a more frequent update of the weights over a single epoch. For instance, taking one patch of size 20×20 at each training iteration corresponds to have $(460/20) \times (700/20) = 805$ weights updates per epoch, allowing to decrease the number of epochs required. Note that this approach is only possible in segmentation and not in classification, since the classes are not defined on a per image basis, but for each of the pixels in the ground truths. However, the choice of padding becomes more relevant than in the whole image training. In fact, assuming a filter of size 5×5 with '`SAME`' padding, we would need to add 4 rows and 4 columns of zeros on each side of the original matrix. On a 460×700 image, this represents an addition of about 1.4% of zeros; on the other hand, the same amount of columns and rows on a 20×20 image represents an increase of pixels of 44%, introducing a huge source of inaccuracy. Our solution to this issue is, once more, to use '`VALID`' padding, fixing the desired output size. For instance, if we target a 5×5 output patch, considering a two layers network, each with a stride of 2 and using filters of dimensions 5×5, we can compute the size at the intermediate layer by inverting relation 2.10:

$$w' = s(w'' - 1) + f = 2(5 - 1) + 5 = 13, \tag{3.1}$$

and at the input:

$$w = s(w' - 1) + f = 2(13 - 1) + 5 = 29. \tag{3.2}$$

This value of the width is greater than the initially choice of $w = 20$, and is not a divisor of the original size of the image. In fact, we choose the original 20×20 crop of the image and add as many pixels as necessary to reach the value of $w = 29$, but instead of adding zeros, we actually utilise the neighbouring pixels values. This means that each patch will overlap partially with its neighbour patches, which is conceptually similar to the overlapping of convolutional filters.

A final note about the dataset concerns the production of the ground truth: we prepared ground truth images by labelling each one by hand, without aid of expert pathologist, introducing, thus, a human error dictated by lack of expertise, which might affect significantly the final accuracy of predictions. Nevertheless, we expect the network to fit our semantic perception of the tissues.

3.2 Architectures and Experiments

The set of architectures utilised in our experiments can be divided into 3 categories: networks not employing max pooling, networks employing max pooling and networks employing max pooling with an additional deconvolutional. The motivations for using deconvolution have been explained in chapter 2; concerning the use or not of max pooling, the choice is generally dictated by the problem settings. The addition of max pooling means to discard the less significant activation map entries after each convolution, removing some information from the output of the precedent layer. Although this deletion may appear to weaken the network, its use is widespread, and we will show that it actually improves the predictions. Schematic representations of each of the three types of networks

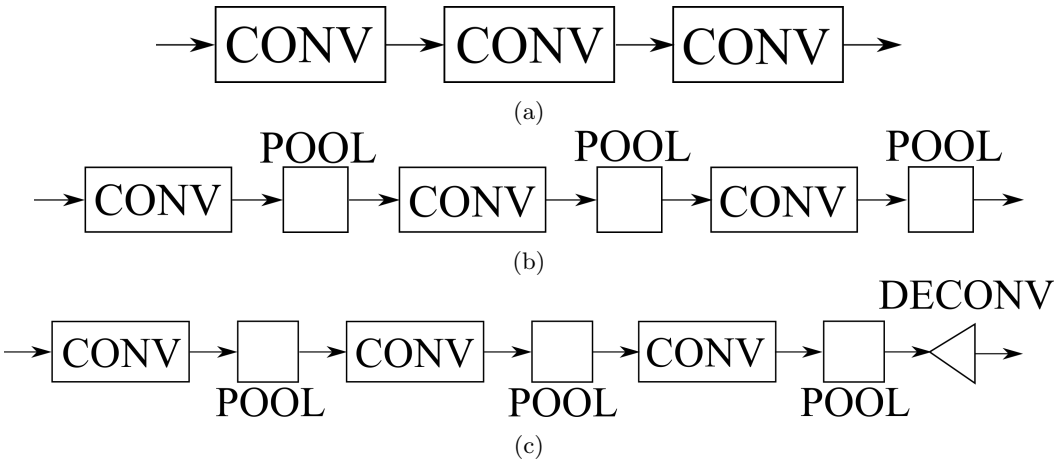


Figure 3.3: Schematic representation of the three architecture categories: a) Only convolutional layers, b) convolutional layers followed by max pooling layers and c) addition of a final deconvolution layer.

are reported in figure 3.3, where 3 convolutional layers have been assumed. Note that, independently of the number of layers, the final output is required to be reduced by a factor of 4 on each dimension, apart from the considerations explained in the previous section. Therefore, appropriate striding has to be chosen; for instance, for a network such as in figure 3.3a the filters will be slid in steps of two in the first and last layer (or any combination of two layers), while for the case with max pooling, this additional operation can be carried out with striding the pooling window. In case of using an additional deconvolutional layer, it is possible to further down-sample by a factor of 2 before applying the new layer, since an up-sampling stride can be used. For instance, in figure 3.3c the 3 convolutional layers can apply an overall down-sampling by a factor of 8 on each dimension, and the last layer will multiply by a factor of 2.

Examples of prediction accuracies for one experiment of each category are shown in figures 3.4-3.6. For each set of experiments, we selected the best performing one, explaining the different choice of parameters used. Most noticeable is the different learning trend between the network without (figure 3.4) and with max pooling (figure 3.5): the accuracy of the former steadily increases over a few hundreds of epochs, until a plateau with some fluctuations is reached, while the second abruptly grows, reaching a more stable plateau, at a larger test accuracy value (68.8% versus 74.6%). This behaviour reflects the exclusion of some pixels after each convolution: the training is more complex with less information available to update the weights, but, once achieved, the result is more stable, as the max pooling "filters" out fluctuations. The case with an additional deconvolution layer is quite

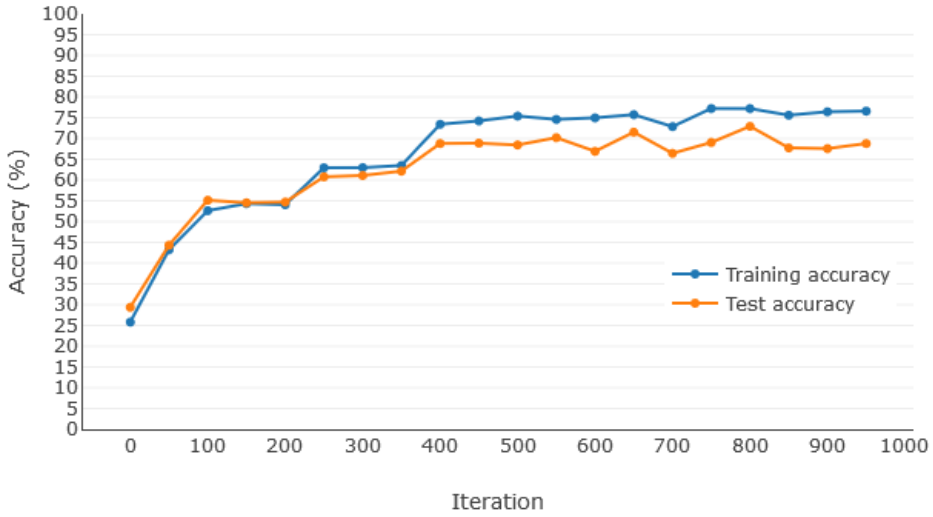


Figure 3.4: Training and test accuracies as a function of the iterations (epochs) for a network of 3 layers, with 16 filters of size 7×7 in the first layer, 32 filters of the same size in the second and 4 filters of size 5×5 in the last. The learning rate and regularisation parameter assume the values $\eta = 0.01$ and $\lambda = 0.5$, respectively

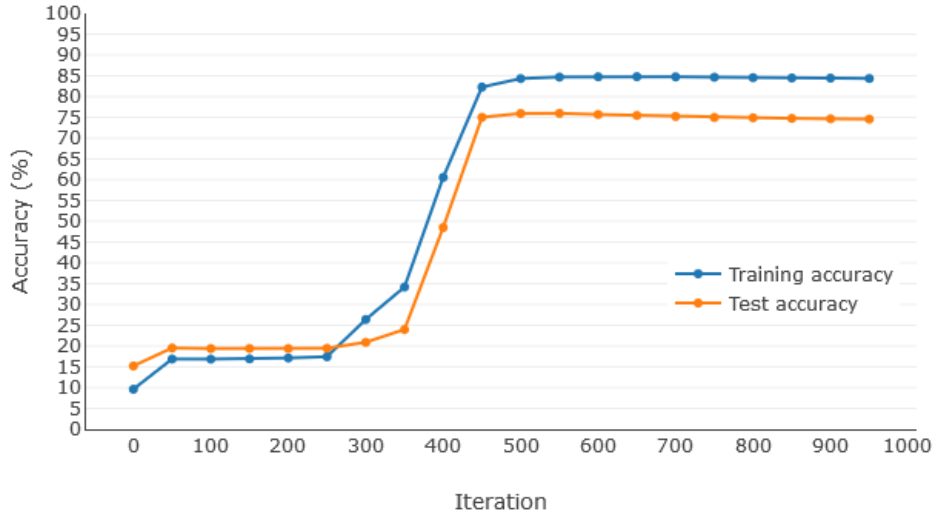


Figure 3.5: Training and test accuracies as a function of the iterations (epochs) for a network of 3 layers, with 32 filters of size 7×7 in the first layer, 64 filters of the same size in the second and 4 filters of size 5×5 in the last. Max pooling with sliding windows of size 2×2 were used after all layers, with strides of 2 for the first and last layer and of 1 for the intermediate. The learning rate and regularisation parameter assume the values $\eta = 0.001$ and $\lambda = 0.1$, respectively

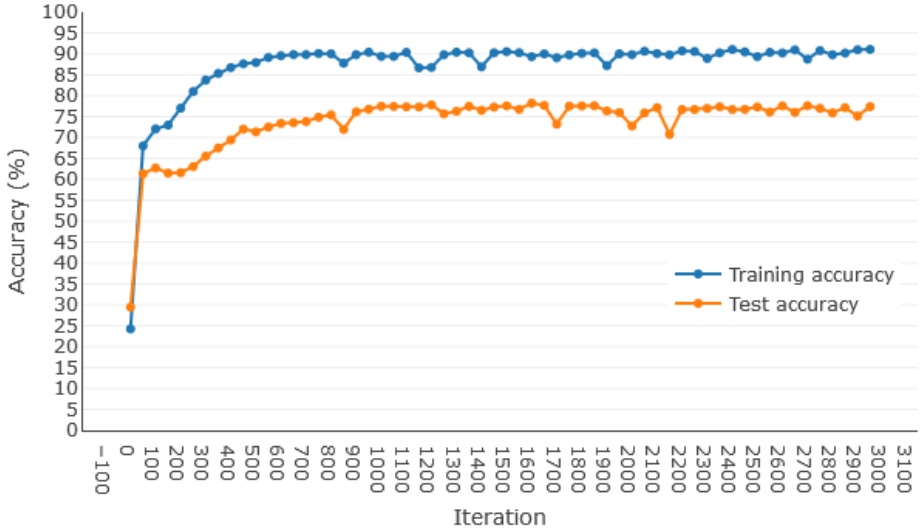


Figure 3.6: Training and test accuracies as a function of the iterations (epochs) for a network of 3 layers, with 32 filters of size 9×9 in the first layer, 64 filters of the same size in the second and 128 filters of size 5×5 in the last. Max pooling with sliding windows of size 2×2 and strides of 2 were used after all layers. An additional deconvolution layer is added, with 4 filters of 5×5, and strides of 2, as in the regular convolutional layers. The learning rate and regularisation parameter assume the values $\eta = 0.001$ and $\lambda = 0.1$, respectively

similar to the latter, although the training phase is a few hundreds of epochs longer, due to the larger number of weights in the network. For the same reason, the over-fitting is larger, reaching values of 93.3% in the training phase, with only little improvement on the test set (75.7%).

All the experiments for each of the categories are summarised in tables 3.1-3.3. In these tables, the filter sizes are reported in the form $f_1/f_2/\ldots/f_n$, with as many entries as are the number of layers in the network and each of them defining the dimensions of the filter as $f \times f$. The number of filters per layer is defined similarly, with $N_1/N_2/\ldots/N_n$ representing a network with n layers and N_i filters in layer i. Inspection of the tables highlights several important points. Consider, for example, table 3.3: we observe little variation of test accuracy when the amount of filters changes,

Input patch	Filters sizes	Filters per layer	Pooling per layer	Epochs	η	λ	Train. Acc.	Test Acc.
460×700	$7/7/5 + 7$	$32/64/128 + 4$	$2/2/2$	1000	10^{-3}	0.1	92.79%	75.60%
460×700	$9/7/5 + 9$	$32/64/128 + 4$	$2/2/2$	1000	10^{-3}	0.1	93.25%	74.57%
460×700	$9/7/5 + 5$	$32/64/128 + 4$	$2/2/2$	1000	10^{-3}	0.1	93.29%	75.70%
20×20	$9/7/5 + 5$	$32/64/128 + 4$	$2/2/2$	100	10^{-4}	0.1	83.06%	69.47%
460×700	$11/9/7 + 5$	$16/32/64 + 4$	$3/3/3$	1000	10^{-3}	0.1	92.38%	72.42%
460×700	$11/9/7 + 5$	$16/32/64 + 4$	$2/2/2$	1000	10^{-3}	0.1	93.10%	74.75%
92×140	$7/7/7 + 5$	$16/32/64 + 4$	$2/2/2$	500	10^{-4}	0.1	87.09%	75.54%

Table 3.1: Parameters used for the set of experiments performed with neural networks with max pooling and a deconvolution layer. All experiments were run with a stride of 1 for the convolutional layers and a stride of two for the max pool layers and the deconvolutional layer. The highlighted row corresponds to figure 3.6.

Table 3.2: Parameters used for the set of experiments performed with neural networks with max pooling and without deconvolution layer. The size of the sliding window in the max pool layers is 2×2 and the convolution strides equal 1 for all entries. The highlighted row represents the data for figure 3.5.

Input patch	Filters sizes	Filters per layer	Pool Strides per layer	Epochs	η	λ	Train. Acc.	Test Acc.
460×700	7/5	32/4	2/2	1000	10^{-3}	0.1	79.08%	73.09%
460×700	7/7/5	32/64/4	2/1/2	1000	10^{-3}	0.1	84.56%	75.27%
460×700	9/7/7/5	16/32/64/4	2/1/1/2	6000	10^{-3}	0.1	87.62%	75.29%
20×20	7/7/5	32/64/4	2/1/2	100	10^{-4}	0.1	87.62%	75.29%
192×140	7/7/5	32/64/4	2/1/2	500	10^{-4}	0.1	82.31%	71.60%

Table 3.3: Parameters used for the set of experiments performed with neural networks with no max pooling or deconvolution layers. The highlighted row corresponds to figure 3.4.

Input Patch	Filters sizes	Filters per layer	Stride per layer	Epochs	η	λ	Train. Acc.	Test Acc.
460×700	7/5/5	16/32/4	2/1/2	1000	10^{-2}	0.5	75.04%	70.21%
460×700	5/5/5	16/32/4	2/1/2	1000	10^{-2}	0.5	71.28%	71.19%
460×700	7/7/5	16/32/4	2/1/2	1000	10^{-2}	0.5	78.04%	71.29%
460×700	7/7/7	16/32/4	2/1/2	1000	10^{-2}	0.5	78.81%	70.77%
460×700	9/7/7	16/32/4	2/1/2	1000	10^{-2}	0.5	79.01%	69.58%
460×700	9/7/5	16/32/4	2/1/2	1000	10^{-2}	0.5	78.12%	70.27%
460×700	9/7/7/5	16/32/64/4	2/1/1/2	1000	10^{-2}	0.5	16.90%	19.56%
460×700	9/7/5	16/64/4	2/1/2	1000	10^{-2}	0.5	78.12%	70.40%
460×700	9/7/5	32/64/4	2/1/2	1000	10^{-2}	0.5	78.41%	70.52%
460×700	9/7/5	32/100/4	2/1/2	1000	10^{-2}	0.5	78.75%	70.31%
460×700	9/7/5	8/32/4	2/1/2	1000	10^{-2}	0.5	77.95%	70.13%
460×700	9/7/5	8/16/4	2/1/2	1000	10^{-2}	0.5	77.74%	69.65%
460×700	9/7/5	8/16/4	2/1/2	1000	10^{-1}	0.5	71.56%	67.61%
460×700	9/7/5	8/16/4	2/1/2	1000	10^{-1}	1	63.48%	64.32%

indicating that increasing the number of weights will, in general, not improve performance. Particularly, increasing the number of weights through the addition of more layers does not achieve larger value of accuracies, as shown in both tables 3.2 and 3.3. This observation is of great importance, since most of the approaches in DL rely on the assumption that "deeper is better", where the depth is defined as the number of layers in a network. However, our experiments with 4 layers networks give accuracies of 19.56% and 75.29% for architectures without and with max pooling, respectively. The accuracies as a function of the iterations for the second one is also reported in figure 3.7, which illustrates how the learning requires much more iterations, without any improvements in accuracy compared to 3 layers networks. On the other hand, the first entry in table 3.2 report the results of a 2 layer network, achieving an accuracy of 73.09%, just slightly smaller than the accuracies of 3 layers networks. Figure 3.8 depicts the accuracies as a function of epochs for this architecture.

In chapter 2 we discussed the issue of choosing appropriate learning rate and regularisation parameter. The tables highlight how this choice is indeed important, but even more relevant is the ratio between the two quantities. In fact, if in equation 2.14 we use $\lambda \gg \eta$, the regularisation term would dominate over the first term, forcing all weights to assume values close to zero, without preserving features connected information. Conversely, if $\lambda \ll \eta$, the presence of the regularisation term would

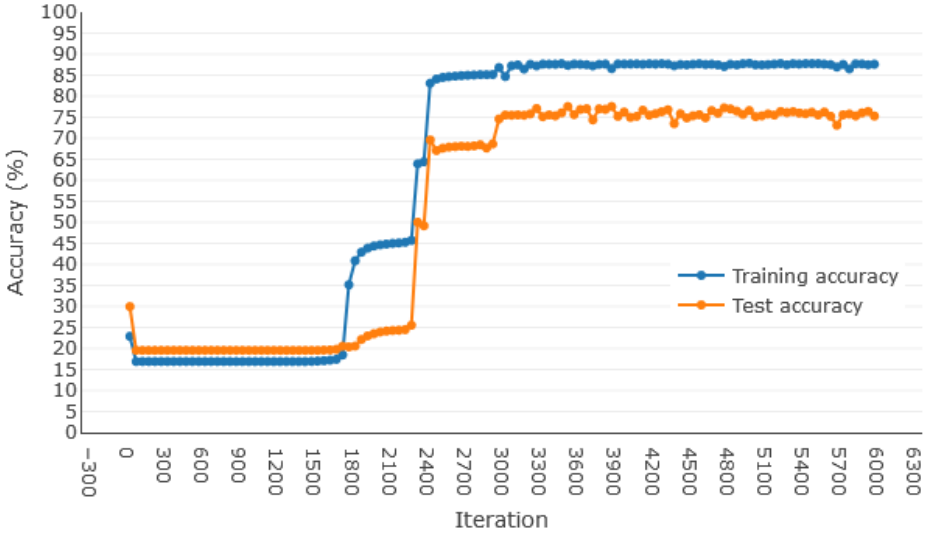


Figure 3.7: Training and test accuracies as a function of the iterations (epochs) for the network with 4 layers reported in table 3.2.

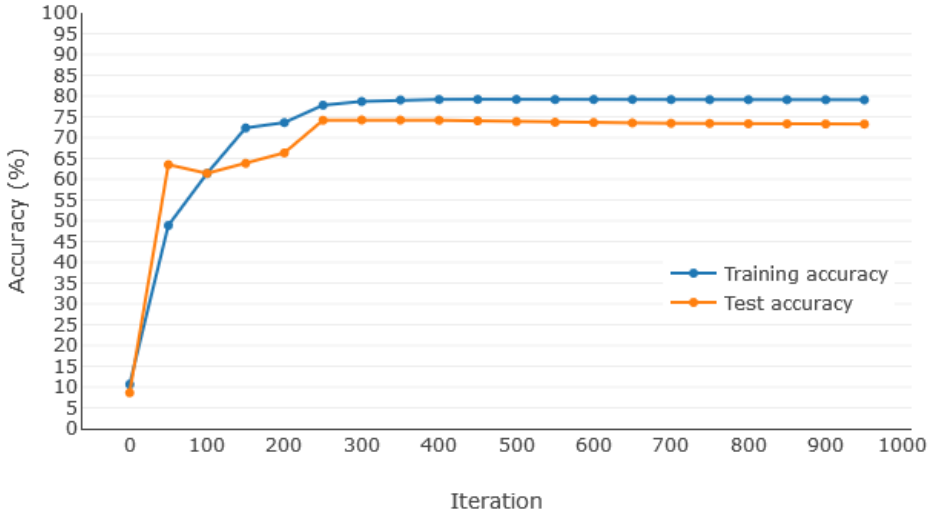


Figure 3.8: Training and test accuracies as function of epochs for a network of 2 layers, with max pooling.

be negligible, and the initial training phase would require significantly more epochs. To understand this statement we consider the plot of the accuracies in figure 3.9 corresponding to the experiment with the 20×20 patch in table 3.2. The number of epochs is much lower than in the previous cases, since, as mentioned in section 3.1, every epoch corresponds to several iterations. Furthermore, being the updates of the weights more frequent, a lower learning rate has been chosen, without changing λ, i. e. the regularisation term is still forcing the weights to assume the lowest possible values while the steps of the gradient are smaller. Nevertheless, the fluctuations at maximum accuracy are more relevant than in the experiments on whole images, showing that the ratio between η and λ is still

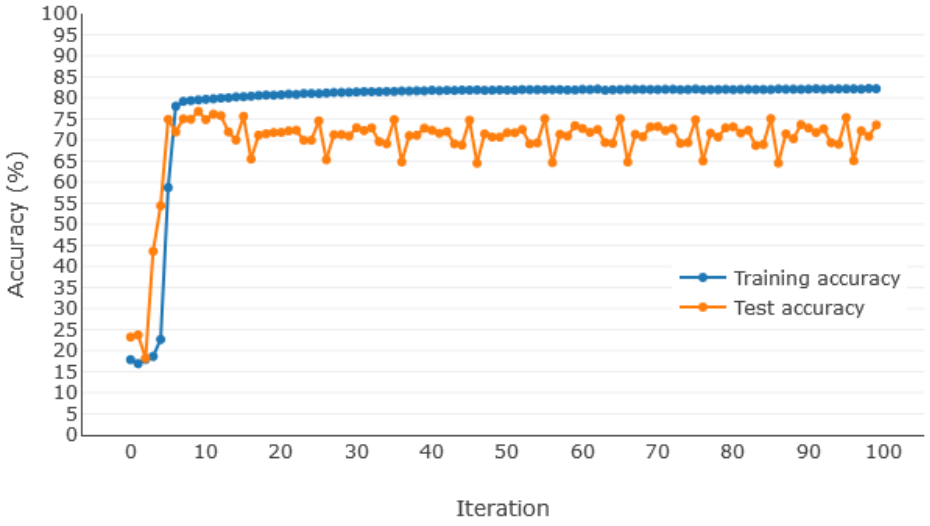


Figure 3.9: Training and test accuracies of a 3 layer network, using as input patches of size 20×20.

large.

The experiments conducted thus far reach accuracies of at most about 75-76%, which is not insignificant, but larger values would be preferred. A limiting factor may be represented by the poor labelling of the images, due to the absence of expert pathologists, or the reduces amount of sample. Therefore, we applied the same methodology to a different type of images, described in A. Janowczyk and A. Madabhushi,[14] and freely available[1]. Their approach identifies only two classes, epithelium and non-epithelium, as illustrated in figures 3.10. More specifically, we used the 3 layers network of

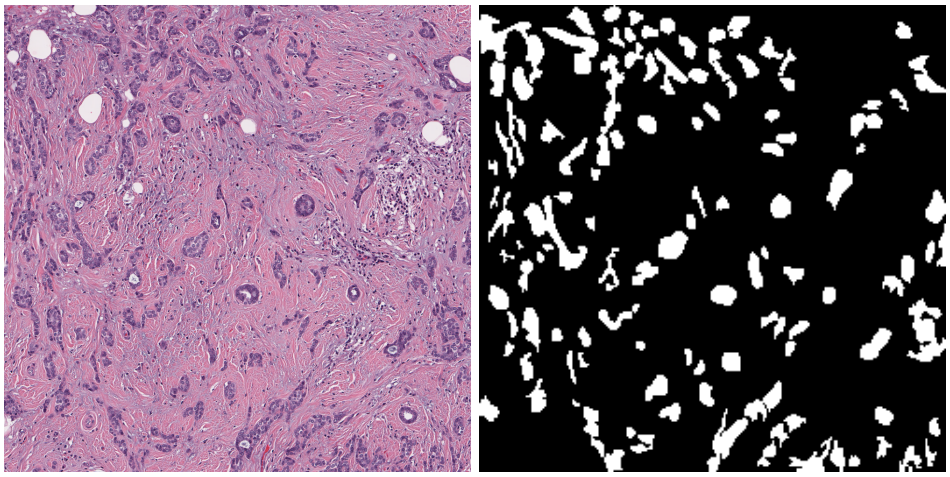


Figure 3.10: Example of image for epithelium segmentation. On the left-hand side is the original image and, on the right, the ground truth image.

table 3.2 which achieved the greatest value of accuracy, and trained it on the new set. The accuracy as a function of epochs is reported in figure 3.11. The final accuracy is larger, 81.7%, and already

[1] http://www.andrewjanowczyk.com/use-case-1-nuclei-segmentation

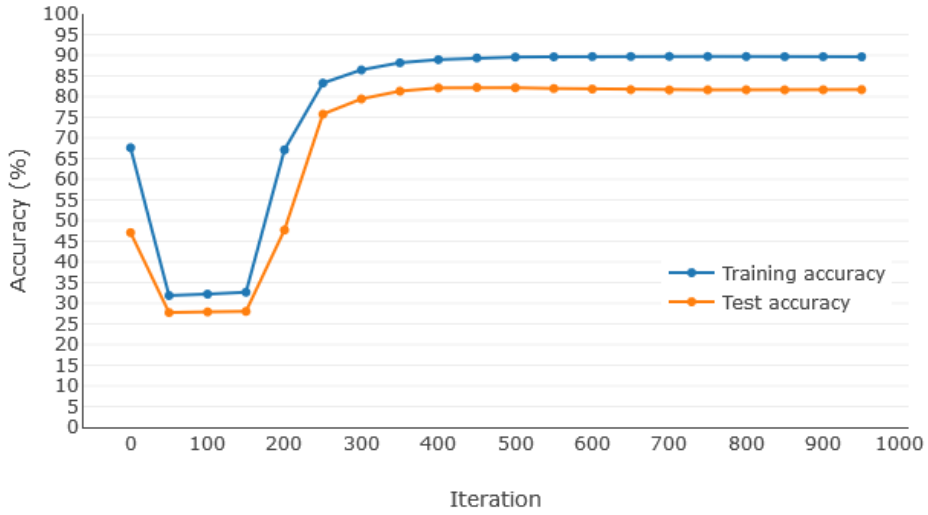
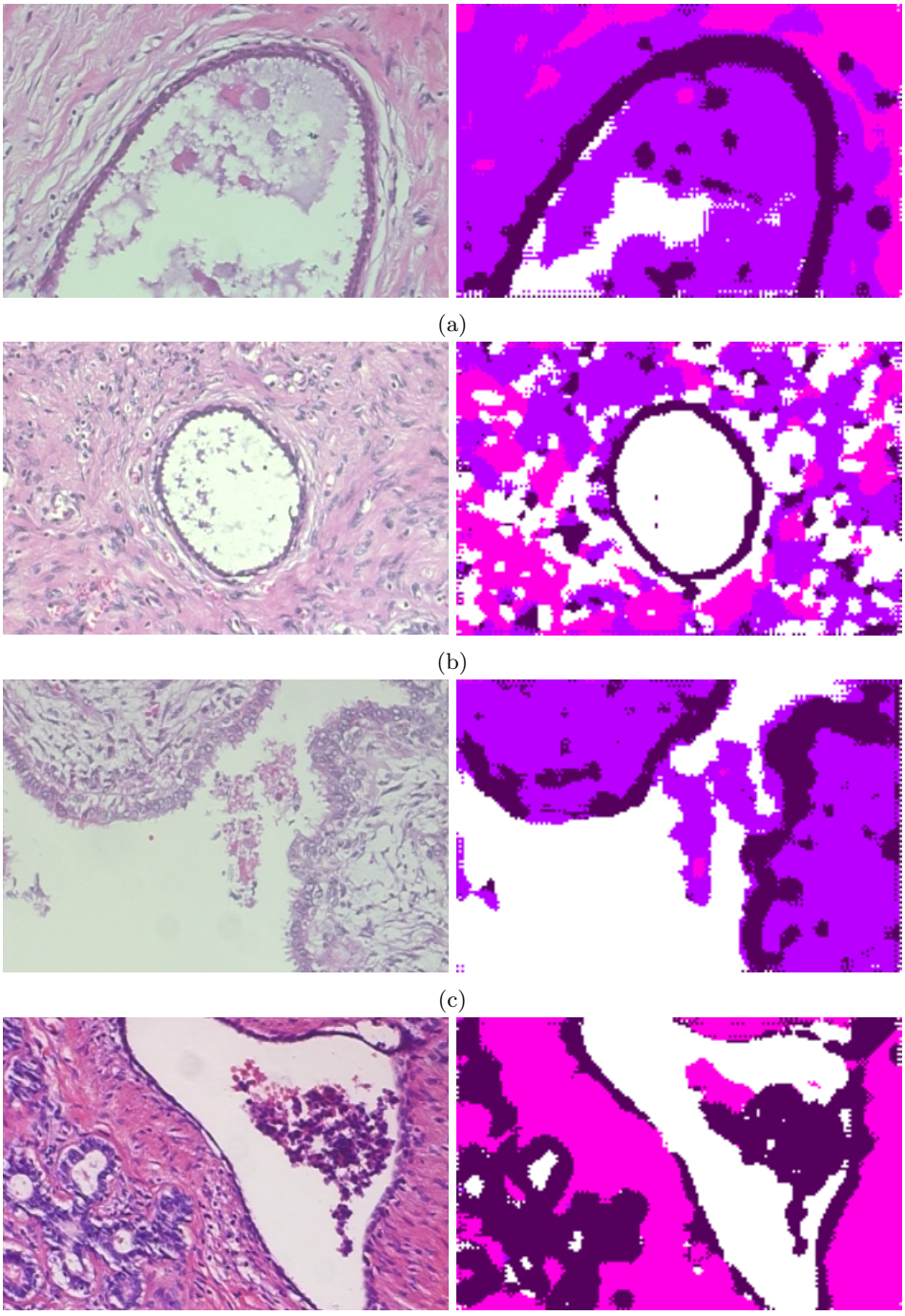


Figure 3.11: Training and test accuracies for the epithelium segmentation problem.

in a more satisfactory range (above 80%). This result shows that our methodology is correct, and the motivations for the lower accuracies with the BreakHis images is related to the poor labelling, or insufficient samples. Note, however, that a two class segmentation is expected to reach larger accuracies, as the task is simpler than with 4 classes.

At this stage, it is worth to inspect the outputs of a network to obtain a visual feedback of a $\sim 75\%$ of accuracy. Figures 3.12 and 3.13 display one sample for each of the tumour types and the labelling obtained using the network which achieved the largest accuracy (table 3.1). The first figure reports the four benign types (adenosis, fibroadenoma, phyllodes tumour and tubular adenoma), while the second one the four malignant types (ductal carcinoma, lobular carcinoma, mucinous carcinoma and papillary carcinoma). The images containing the labels show that the overall geometric features of the histological slide is learned, although the classes are only roughly correct. Moreover, on the boundaries between two classes, some artifact is present, which might affect the predictive power in the second step. However, some pre-processing techniques can be used, which will be discussed in the next chapter.



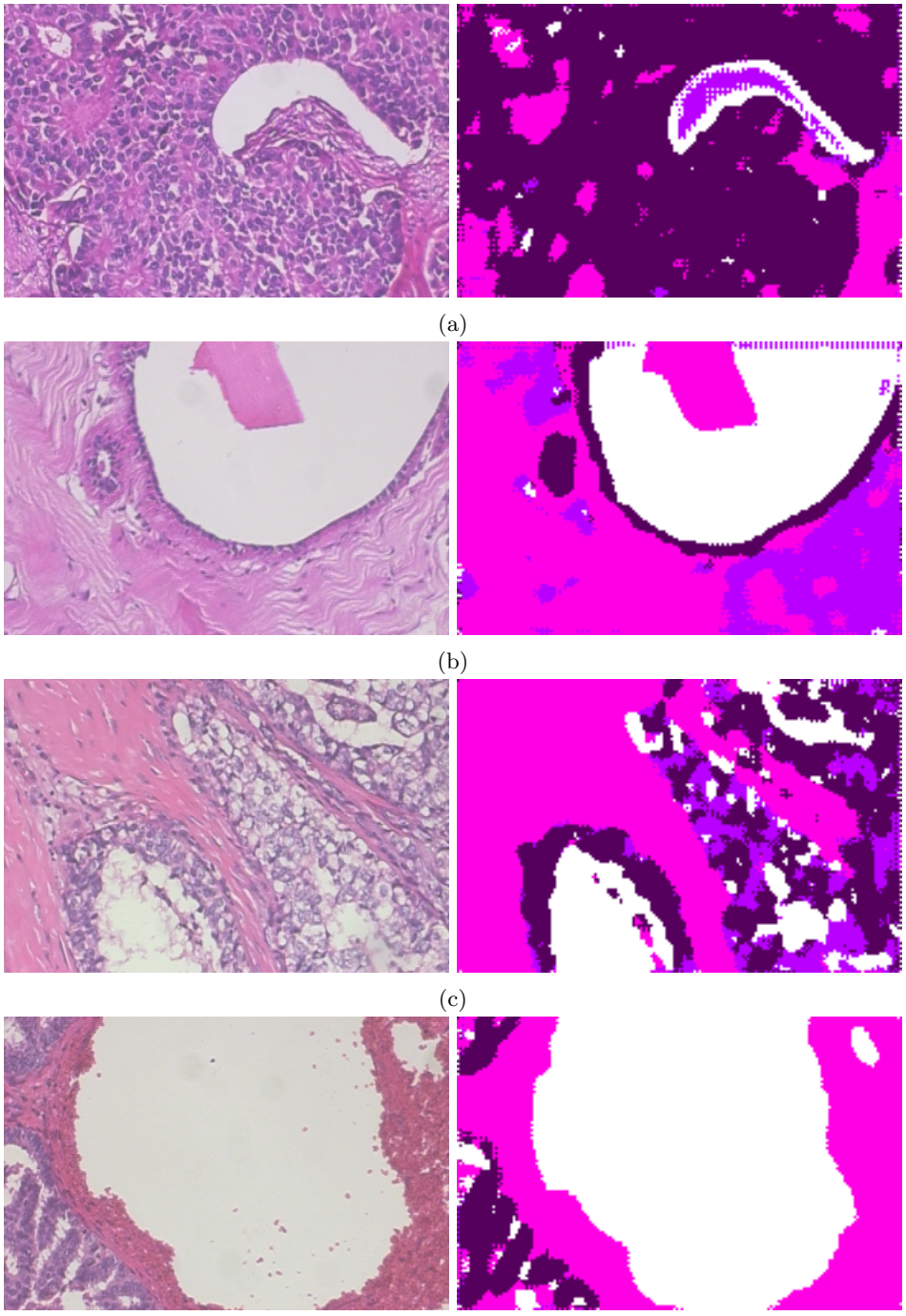
(a)

(b)

(c)

(d)

Figure 3.12: Original images and respective labelling performed by the trained network, for benign tumour types: a) adenosis, b) fibroadenoma, c) phyllodes tumour and d) tubular carcinoma.



(a) (b) (c) (d)

Figure 3.13: Original images and respective labelling performed by the trained network, for malignant tumour types: a) ductal carcinoma, b) lobular carcinoma, c) mucinous carcinoma and d) papillary carcinoma.

Chapter 4

Classification Experiments

The results of the classification problems, on both original and segmented images, are illustrated in this chapter. Classification of the original set is performed to compare the end-to-end approach to the two-step approach, as outlined in chapter 2.

All experiments have been executed on a computer with an NVIDIA® GPU, more precisely a commercial Geforce® GTX 980, limiting slightly the performance in both speed and memory availability. In fact, classification networks are quite demanding and often higher quality GPUs, such as the NVIDIA Quadro® series, are better suited to the problem.

4.1 Architectures and Experiments

The images used for the classification experiments have been drawn from the set of 100x magnification images of the BreakHis database. Since we require each class to have the same amount of samples as the remaining ones, we selected the breast cancer type with the least number of images (adenosis) and chose the same quantity for all the other categories. In practice, 113 images were selected for each class, resulting in a total of 904 images for classification, to be split into training and test samples. Contrarily to the segmentation problem, each image corresponds to a single label and, thus, the number of training samples is much lower compared to the previous experiments. Therefore, instead of 8 classes, we decided to classify the images into benign and malignant, reducing the number of classes to 2. Finally, we segmented the same 904 images to be used in the second stage of the two-stage approach.

Figure 4.1 shows an example of accuracies for end-to-end classification. This graph was obtained using a network of 6 layers, with 8, 16, 32, 64, 128 and 256 filters respectively in the convolutional layers, 256 neurons in the first fully connected layer and 2 in the second and last one. Max pooling has been used after each convolution with strides of 2, down-sampling the original images of dimensions 460×700 pixels by a factor of $2^6 = 64$, before feeding them into the fully connected layers. After some trial and error, we chose $\eta = 10^{-4}$ and $\lambda = 10^{-1}$. The plot shows that the network has a strong tendency to over-fit, with a rather low test accuracy ($\sim 66.7\%$). Analogously, figure 4.2 reports the accuracies for the case of segmented images. It is important to note that the colours in these images are a mere visual representation of the different labels, and, thus, they have to be converted to numerical values. For the sake of symmetry, we defined the four classes as $\{-3, -1, 1, 3\}$ for background, epithelium, epi-stroma and stroma, respectively. The final test accuracy is excessively low ($\sim 53.0\%$), considering also that a random guess on a 2 class problem would result in a 50% of accuracy, on average. Therefore, some attempts have been made to increase the predictive power of the network, in line with the explanation of section 2.2.

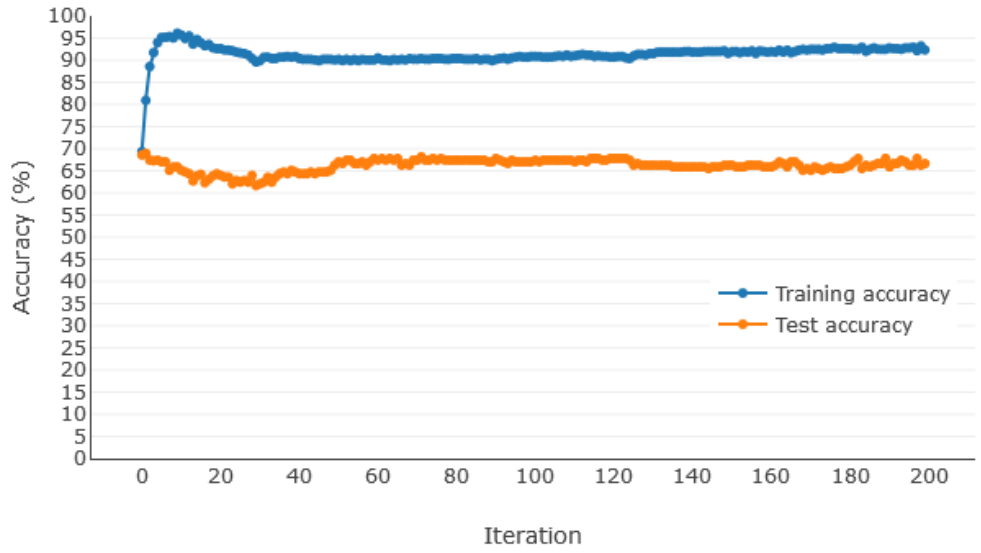


Figure 4.1: Training and testing accuracies as a function of epochs for the classification of original BreakHis images.

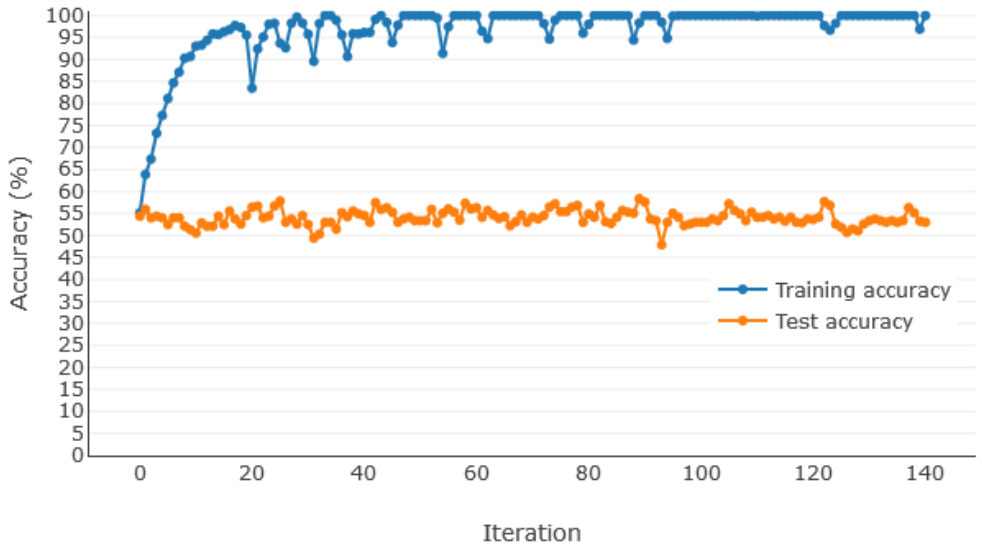


Figure 4.2: Training and test accuracies as a function of epochs for the case of classification of segmented images.

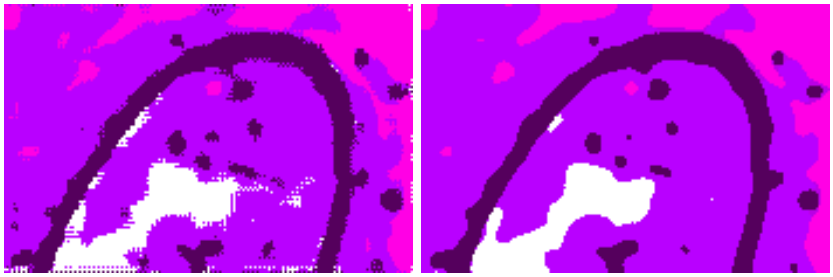


Figure 4.3: Example of segmented image output by the segmentation stage without (left-hand side) and with (right-hand side) the application of a 3×3 median filter.

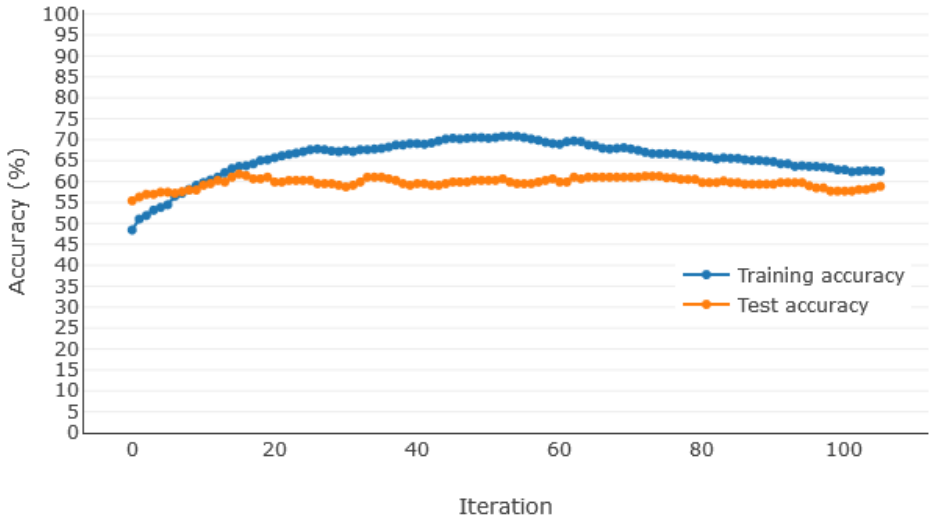


Figure 4.4: Training and test accuracies for the same network as in figure 4.2, after applying a 3×3 median filter to the segmented images.

Apart from neural network related techniques, we also applied some pre-processing to the segmented images, to get rid of the noisy output. Figures 4.3 demonstrate the result of the application of a 3×3 median filter to an image output of the segmentation step. The smoother labelling is expected to simplify the learning process of the network. In fact, figure 4.4 reports the accuracies for the same network of figure 4.2, trained on the pre-processed images. Although not significantly, the final value is somewhat larger than a random guess ($\sim 58.88\%$) and closer to the value obtained on the original images classification, while the over-fitting is drastically reduced. On the other hand, figure 4.5 illustrates the accuracies after applying a 20% of dropout, as explained in chapter 2. The final values are completely random, for both training and test curves, showing an evident case of under-fitting.

The small values of accuracies obtained in this second part of our experiments may not appear very meaningful, but it is interesting to note that the two methods achieve comparable results. Unfortunately, both methods are very demanding in terms of memory requirement, e. g. attempts to double the number of filters in each layer were unsuccessful because of lack of memory on our commercial Geforce GTX. Most likely, in this second scenario of classification, increasing the number of layer might actually improve the performances of the networks, but better computational resources

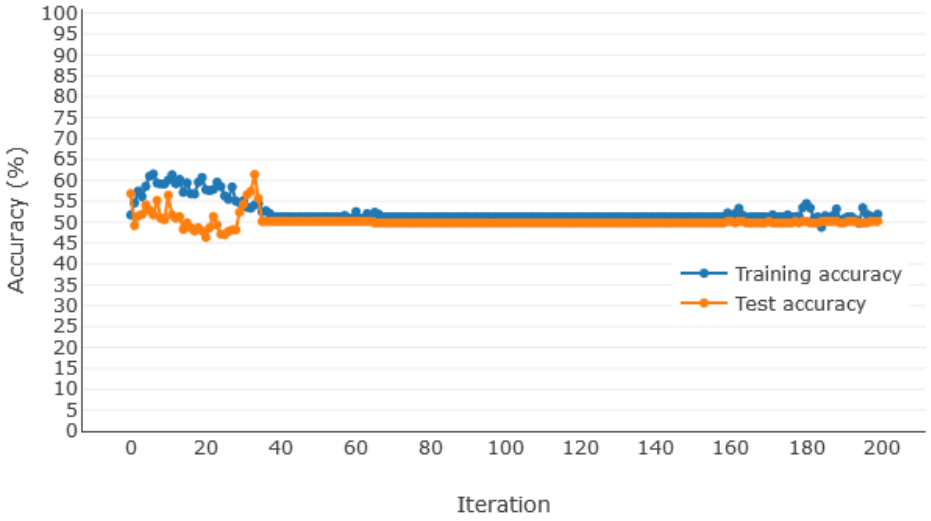


Figure 4.5: Training and test accuracies for the same network as in figure 4.2, using 20% of dropout.

would have been necessary. Furthermore, the set of 904 images is also quite small, and a larger database would allow to train the network to larger accuracies. Nevertheless, we might conclude that the two methods reach indeed similar results, showing that most of the relevant visual features are preserved during segmentation into four classes. Moreover, the similar approach used in J. DeFauw et al.[16] confirms the solid grounds of our idea.

Chapter 5

Conclusions

We proposed, in this work, a two-stage segmentation plus classification approach to assist medical staff in the prediction of breast cancer typologies from histopathological images. Our method aimed to give some insight about the visual features relevant to the detection of these types of tumours, in contrast with *black box* end-to-end classification schemes, very widespread among the machine learning experts.

For this purpose, we engineered convolutional neural networks to segment histological images of breast tissue into four classes we deemed more relevant, and exploited different techniques, such as max pooling and deconvolution, in the design of the architectures. We were able to understand the most relevant parameters in such design, and, most importantly, to find some architectures which were able to give significant test accuracies, despite the absence of expert pathologist in the ground truth labelling process and the limited amount of samples. Moreover, we show that increasing the number of layers in a so called deep network is not, in general, a synonym of better performance with respect to 2- or 3- layers network, in stark contrast with the widespread tendency to make architectures with as many layers as possible.

The somewhat shorter second part focuses on applying a classification scheme to the segmented images obtained in the first part. Although the final values of accuracies are very low for classification of tumour types into benign and malignant, they are comparable with an analogous classification scheme applied directly to the original images. Therefore, the limited performance is most likely due to the lack of better computational resources and to the low number of sample available in the BreakHis dataset.

Despite the results, in particular in the second part, appear to be not very appealing, they give ground to future possibilities into the same direction, also supported by recent works, such as the one of J. DeFauw et al.,[16] in line with our idea and published during the development of the present project.

Bibliography

[1] World Health Organization. *WHO position paper on mammography screening.* World Health Organization (2014).

[2] M. Malatesta. *Histological and Histochemical Methods - Theory and practice.* Eur. J. Histochem., **60(1)** (2016).

[3] F. A. Spanhol, L. S. Oliveira, C. Petitjean et al. *A Dataset for Breast Cancer Histopathological Image Classification.* IEEE Trans. Biomed. Eng., **63(7)**:1455–1462 (2016).

[4] R. Szeliski. *Computer vision : algorithms and applications.* Springer, London [etc.] : (2011).

[5] B. Zhang. *Computer vision vs. human vision.* In *9th IEEE Int. Conf. Cogn. Informatics*, 3–3. IEEE (2010).

[6] A. Krizhevsky. *Learning Multiple Layers of Features from Tiny Images* (2009).

[7] A. Madabhushi and G. Lee. *Image analysis and machine learning in digital pathology: Challenges and opportunities.* Med. Image Anal., **33**:170–175 (2016).

[8] D. A. Forsyth and J. Ponce. *Computer vision : a modern approach.* Pearson Education, Boston [etc.] :, 2nd. ed. edition (2012).

[9] G. Lee, R. Sparks, S. Ali et al. *Co-Occurring Gland Angularity in Localized Subgraphs: Predicting Biochemical Recurrence in Intermediate-Risk Prostate Cancer Patients.* PLoS One, **9(5)**:e97954 (2014).

[10] F. Xing and L. Yang. *Robust Nucleus/Cell Detection and Segmentation in Digital Pathology and Microscopy Images: A Comprehensive Review.* IEEE Rev. Biomed. Eng., **9**:234–63 (2016).

[11] Z. Han, B. Wei, Y. Zheng et al. *Breast Cancer Multi-classification from Histopathological Images with Structured Deep Learning Model.* Sci. Rep., **7(1)**:4172 (2017).

[12] A. Cruz-Roa, H. Gilmore, A. Basavanhally et al. *Accurate and reproducible invasive breast cancer detection in whole-slide images: A Deep Learning approach for quantifying tumor extent.* Sci. Rep., **7(1)**:46450 (2017).

[13] B. Ehteshami Bejnordi, M. Balkenhol, G. Litjens et al. *Automated Detection of DCIS in Whole-Slide H&E Stained Breast Histopathology Images.* IEEE Trans. Med. Imaging, **35(9)**:2141–2150 (2016).

[14] A. Janowczyk and A. Madabhushi. *Deep learning for digital pathology image analysis: A comprehensive tutorial with selected use cases.* J. Pathol. Inform., **7(1)**:29 (2016).

[15] W. Samek, T. Wiegand and K.-R. Mller. *Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models.* ITU Journal, **1**:1–10 (2017).

[16] J. De Fauw, J. R. Ledsam, B. Romera-Paredes et al. *Clinically applicable deep learning for diagnosis and referral in retinal disease.* Nat. Med., **24**:1342–1350 (2018).

[17] A. Krizhevsky, I. Sutskever and G. E. Hinton. *ImageNet classification with deep convolutional neural networks* (2012).

[18] W. S. McCulloch and W. Pitts. *A logical calculus of the ideas immanent in nervous activity.* Bull. Math. Biophys., **5(4)**:115–133 (1943).

[19] J. Hertz, A. Krogh and R. G. Palmer. *Introduction to the theory of neural computation.* Addison-Wesley Pub. Co (1991).

[20] J. Long, E. Shelhamer and T. Darrell. *Fully Convolutional Networks for Semantic Segmentation* (2015).

[21] D. P. Kingma and J. Ba. *Adam: A Method for Stochastic Optimization.* ArXiv e-prints (2014).

[22] K. I. Bland, V. S. Klimberg, E. M. Copeland et al. *The breast : comprehensive management of benign and malignant diseases.* Elsevier (2018).