POLYTECHNIC UNIVERSITY OF TURIN

MASTER THESIS

Transfer Learning and Data Augmentation for Semantic Segmentation in Histopathology

Author:
Giorgia CANTISANI

Supervisor:
Elisa FICARRA
Santa DI CATALDO
Francesco PONZIO

A thesis submitted in fulfillment of the requirements for the Master Degree in

Biomedical Engineering

DAUIN
Department of Control and Computer Engineering

July 5, 2018

Declaration of Authorship

I, Giorgia CANTISANI, declare that this thesis titled, "Transfer Learning and Data Augmentation for Semantic Segmentation in Histopathology" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a Master's degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

"La donna è stata bloccata per secoli. Quando ha accesso alla cultura è come un'affamata. E il cibo è molto più utile a chi è affamato rispetto a chi è già saturo."

Rita Levi-Montalcini

Abstract

The segmentation of Region Of Interests (ROIs) in high-resolution histopathological images is a task with high clinical relevance which requires large amounts of reading time from pathologists and suffers from many considerable problems, such as *inter* and *intra-reader variability*. Thus, its automation is desirable because it would hugely reduce the workload of the clinicians, while at the same time reducing the subjectivity of the diagnosis. However there are still open issues, like the amount of data required for training and testing algorithms, the computational load when dealing with high resolution images and, most of all, the large variability in the analyzed data (mainly due to different cell types, cell densities, stains, magnification levels, and so on).

The segmentation of ROIs can be expressed as a Semantic Segmentation problem for which Deep Learning models are proven to be particular suitable. However building these models requires a huge amount of annotated training data. To overcome this limitation, Transfer Learning and Data Augmentation have been proposed as possible solutions.

This thesis shows how Transfer Learning and Data Augmentation can be leveraged for Semantic Segmentation of histopathological images.

The UNet convolutional-deconvolutional neural network is trained using the Camelyon16 Dataset (H&E breast cancer Whole-Slide Images). Whole-Slide Images (WSIs) are first preprocessed (tissue regions segmented from fatty and white background areas and then stain-normalized) and then fed as tiles to the CNN. The outputs are probability maps, whose pixel intensity values express the probability of being part of a cancer metastasis. In a second phase, a fine-tuning of the pre-trained UNet is performed on the GlaS (H&E glands images) and ISBI12 (Drosophila first instar larva ventral nerve cord ssTEM) datasets in order to understand whether the trained model can be used for segmenting other types of images for which the available datasets are not sufficient. Fine Tuning was aided by a Data Augmentation pre-processing.

Results are very promising and, to my knowledge, such approach represents a novelty not only in the biomedical field, but also in the computer vision one.

Acknowledgements

Sento l'esigenza di ringraziare innanzitutto i miei genitori, Claudio e Stefania, per tutto il sostegno e l'aiuto che mi hanno dato in questi anni. Vorrei dire loro che non ho mai dato per scontata l'opportunità di studiare che mi hanno regalato e che gliene sarò per sempre grata. Li ringrazio perchè hanno sempre fatto tutto questo senza mostrarmi la fatica e i sacrifici che ne sono derivati.Voglio che sappiano che quei sacrifici e quella fatica non sono passati per nulla inosservati.

Ringrazio Diego per avermi letteralmente trascinato con la sua spensieratezza e il suo entusiasmo fino a questo traguardo senza farmi sentire la fatica e la paura.

Ringrazio Francesco e Santa per avermi lasciato libera di scegliere, sbagliare e sperimentare senza impormi percorsi stabiliti. Questo mi ha fatto sentire doppiamente orgogliosa del mio lavoro e più consapevole delle mie capacità.

Ringrazio anche le professoressa Elisa Ficarra e il Dauin (Dipartimento di Automatica e Informatica del Politecnico di Torino) per le risorse computazionali messe a disposizione con il progetto hpc@polito (http://www.hpc.polito.it).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Semantic Segmentation in Digital Pathology

The identification of regions of cells and masses according to some medical criteria are typical segmentation tasks performed on high-resolution histopathological images. Such a task has a seminal clinical relevance and requires large amounts of reading time and effort from pathologists. Traditionally, the pathologist bases his analysis only on experience and the examination is done manually with the aid of a microscope. Therefore, this process can suffer from many considerable problems, such as *inter* and *intra-reader variability*. Therefore, an automation of this process would hugely reduce the workload of the pathologists while at the same time reducing the subjectivity of the diagnosis. However there are still open issues like the amount of data required for building and testing analysis algorithms, the computational load when dealing with high-resolution images and, most of all, the large variability in the analyzed data. The latter factor mainly due to different cell types, cell densities, stains, magnification levels, and so on.

When the segmentation attempts to partition the image into semantically meaningful parts *and* classify each of them into one pre-determinated classed, it is usually referred as *semantics* segmentation. For such task, Deep Learning convolutional models are proven to be really suitable and nowadays they represent the state-of-the-art approach. However it is well known that their performance is proportional to the number of the *training* data which are by definition labeled and for certain applications, especially in the medical field, they are really difficult and expensive to collect. In order to overcome the *small dataset* issue, techniques named *Transfer Learning* and *Data Augmentation* have been proposed as possible solution.

1.2 Transfer Learning for Semantic Segmentation

Within this thesis I evaluated the possibility of applying Transfer Learning and Data Augmentation in the field of Semantic Segmentation of histopathological images. In particular I focused on the problem of Transfer Learning applied to Semantic Segmentation which, to my knowledge, represents a novelty not only in the biomedical field, but also in more broad computer vision one.

In machine learning, Transfer Learning is the problem of storing knowledge gained while solving one task and applying it to a different but related one. For instance, the knowledge gained for recognize cats can be used when trying to recognize a tigers. It was widely explored and studied, however these techniques are often limited to simple classification tasks [1] (e.g. exploiting networks pre-trained on ImageNet). According to [1] there are some works on learning visual attention, i.e. networks that can adaptively focus on salient part of an image or video for

performing different tasks, such as object recognition, object tracking, caption generation, image generation, and so on. In [2] a transfer learning approach for object detection is proposed, but the transfer of knowledge is between categories. Only [1] proposed a specific framework on transfer learning applied to semantic segmentation: they tried to exploit segmentation annotations from different categories to guide segmentations with weak annotations.

1.3 Formalization of the problem

The problem can be formalized as follow: try to build a complete transfer learning pipeline for semantic segmentation.

First, a UNet convolutional-deconvolutional neural network is implemented in `Keras` (`Tensoflow/Theano` backend) and trained using the Camelyon16 dataset (H&E breast cancer Whole-Slide Images) on a segmentation task. The dataset was properly chosen because of his size and well-labeled ground truth masks. Whole-Slide Images (WSIs) are first preprocessed (tissue regions segmented from fatty and white background areas and then stain-normalized) and then fed as single tiles to the CNN. The outputs are probability maps, whose pixel intensity values express the probability of being part of a cancer metastasis.

In a second phase, the transfer learning is performed through fine-tuning: inner and different layers of the pre-trained UNet are tuned for the GlaS (H&E glands images) and the ISBI12 (ssTEM Drosophila first instar larva ventral nerve cord images) datasets. This is done in order to understand if the trained model can be used for segmenting other types of images for which the available datasets are not sufficient. Fine Tuning is aided by a Data Augmentation preprocessing.

In the third phase, the results of the fine tuning are analyzed taking into account the number of layers which were fine-tuned, the type of data augmentation applied to the datasets and some preprocessing parameters.

1.4 Thesis organization

The rest of the thesis is organized as follows:

- Chapter 2: *Deep Learning and Semantic Segmentation*. In this chapter Deep Learning and Convolutional Neural Networks are introduced as State of the Art solution to the problem of Semantic Segmentation. Data Augmentation and Transfer Learning are presented as enhancing techniques for improving the performances and for reducing the required computational resources.
- Chapter 3: *State of the Art*. In this chapter the State of the Art of Convolutional Neural Networks for Digital Pathology and Microscopy is presented. A particular attention is given to the two publicly available datasets used for the experimental part of this thesis.
- Chapter 4: *Methods and Experimental Set-up*. In this chapter the experiment setting is explained step by step starting from the preprocessing stage up to the fine-tuning phase.
- Chapter 5: *Results Analysis*. In this chapter the results of the experiment are analyzed and interpreted.

- Chapter 6: *Conclusions and Future Work*. In this chapter, the most important results are summarized, limits of this work are presented and some indications are stated for future improvement.

Chapter 2

Deep Learning and Semantic Segmentation

2.1 Deep Learning

Over the past few years, Deep Learning has been one of the most innovative and surprising breakthrough technologies in the science world (most of the information of this chapter about Deep Learning has been taken from [3] where you can find a complete review). The major fields of the application of Deep Learning are Computer Vision, Image Processing, Speech Recognition, Advancement in Natural Language Processing and, potentially, any other field.
In the academic environment there is still some resistance to this technology because it is still not clear how it really works; however the record holders on ImageNet and Semantic Segmentation are Convolutional Neural Networks (CNNs). Moreover, CNNs are already part of commercial products like Google's and Microsoft's speech recognition systems.

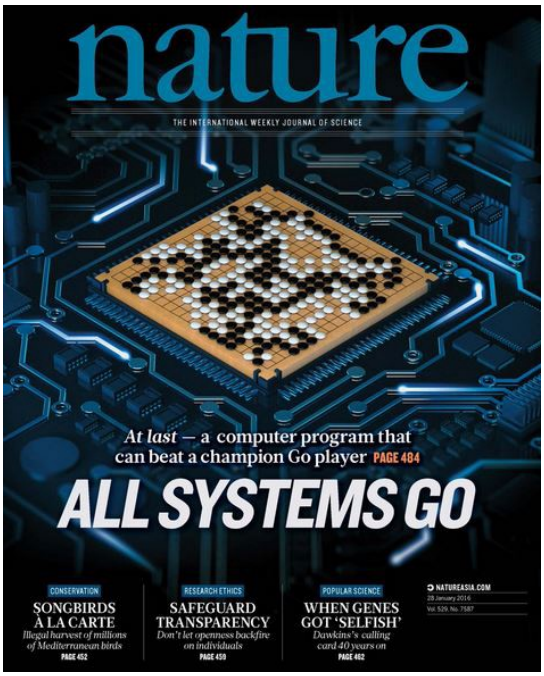


FIGURE 2.1: AlphaGo paper published on Nature (nr. 16961).

Deep Learning was first conceived for image classification tasks to overcome the limitations of the classical Neural Networks (NNs). In fact, NNs present a high training cost because each neuron can be considered as a regression algorithm. Training

the entire NN means training all the interconnected regressions. Moreover, when the number of hidden layers increases, the NN is more difficult to train because in the back-propagation the gradient gets progressively more dilute which means that the correction below the top layers is minimal. Last but not least there is the problem of local optima due to the random initialization.

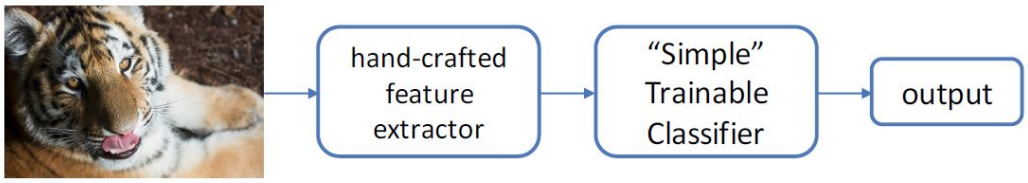


FIGURE 2.2: Traditional ML pipeline [4].

The solution to these problems is Deep Learning, i.e. learning multiple levels of representation. A Deep Learning network presents a cascade of many hidden layers of locally connected units, for feature extraction and transformation. The hidden layers learn multiple levels of representations that correspond to different levels of abstraction. We can say that the levels form a hierarchy of concepts.

But where does the idea of a deep structure come from? We can say that Deep Learning is inspired principally by nature: the brain has a deep architecture and cognitive processes seem to be deep as well. The mammalian visual cortex, for example, presents a sequence of areas each of which contains a representation of the input, and signals flow from one to the next. Moreover, it seems that representations in the brain are both distributed and purely local, i.e. they are sparse: about 1% of neurons are active simultaneously in the brain. Furthermore, analyzing cognitive processes, it seems that humans organize their ideas and concepts hierarchically: they first learn simpler concepts and then compose them to represent more abstract ones.

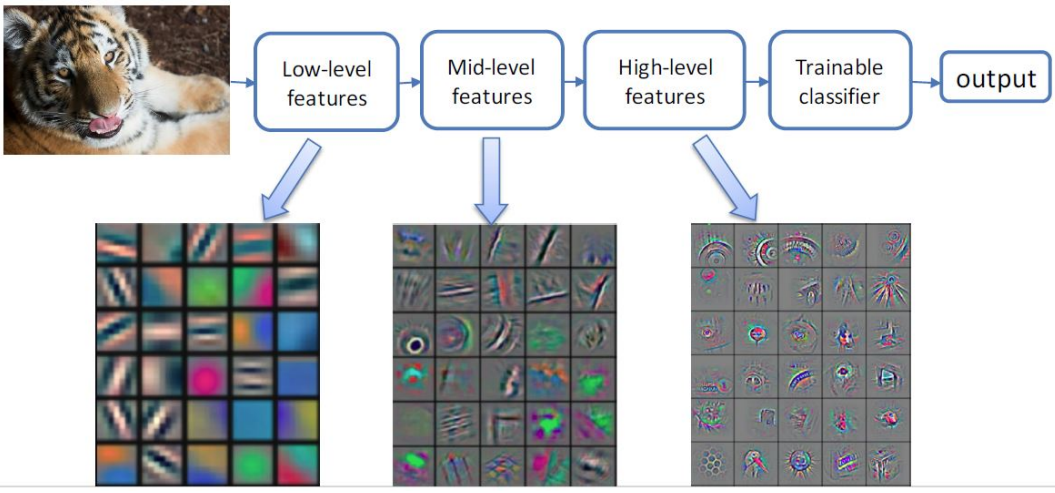


FIGURE 2.3: Deep Learning pipeline [4].

In addition to the biological aspects there are also some technical considerations. First, shallow architecture (like SVM, KNN, etc.) are not so efficient as Deep Learning when the number of parameters in the training phase is too large. Second, traditional classifiers use hand-crafted features which are usually highly dependent on

one application and cannot be transferred easily to other applications.
The classical Machine Learning (ML) pipeline is reported in Figure 2.2.

Deep learning, instead, seeks to learn rich hierarchical representations (i.e. features) automatically through multiple stages of feature learning processes. The result is a hierarchical representation with an increasing level of abstraction where each stage is a kind of trainable nonlinear feature transform. For example, in the case of image recognition, the hierarchical sequence can be: pixel-edge-texture-motif-part-object; while in the case of text recognition can be: character-word-word group-clause-sentence-story. The typical Deep Learning pipeline is reported in Figure 2.3.

2.1.1 Convolutional Neural Networks (CNNs)

Convolutional Neural Network (CNN) is one of the most famous model of Deep Learning and was inspired by the neurophysiological experiments conducted by Hubel & Wiesel in 1962 [5].Originally built for image classification, CNNs try to replicate the mammalian visual cortex which contains a complex arrangement of cells, that are sensitive to small sub-regions of the visual field called receptive fields. These cells act as local filters over the input space and are well-suited to exploit the strong spatially local correlation present in natural images. In particular we can distinguish two types of cells:

- simple cells which respond maximally to specific edge-like patterns within their receptive field;
- complex cells which have larger receptive fields and are locally invariant to the exact position of the pattern.

According to these biological observations, CNNs can be considered a special type of NN whose hidden units are only connected to local receptive field. Thanks to this characteristic, the number of parameters needed by CNNs is much smaller and the input can have very high dimension.

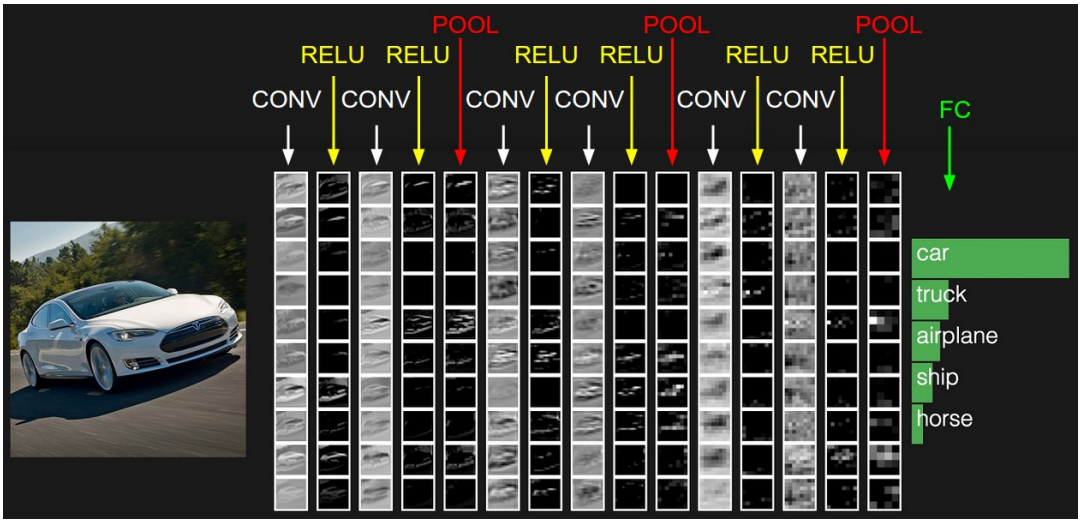


FIGURE 2.4: VGG network [3]

Conclusively a CNNs can be viewed as a list of layers that transforms the input data into an output class/prediction. Each layer acts as feature extractor:

- low-level layers extract local features;

- high-level layers extract global patterns.

In particular the layers can be distinguished in four different types [3]:

- convolutional layers (CONV);
- non-linear layers (RELU);
- pooling layers (POOL);
- fully connected layers (FC).

Usually these types of layers are interchanged to build the classical CNNs architecture: first we have the input layer which is followed by multiple iterations of the sequence CONV-RELU-POOL and at the end we can find the FC. In Figure 2.4 we can see an example of CNN architecture.

Convolutional Layers

A convolutional layer is the most characteristic layer of a CNN. It consists of a set of filters each of which covers a spatially small portion of the input data. Each filter is convolved over the entire image in input in order to produce a multidimensional feature map. Usually there are multiple feature maps, one for each convolution operator. In this way the network will learn the parameters of the filters that activate when they see specific type of features in a specific area of the input image.
The basis of convolutional layers is the 2D convolutional operation. This operation in based on the convolutional kernel which is a 2D array of learnable parameters, obviously smaller than the input image, that slides with overlapping over the entire input image like in Figure 2.5.

$$s[i,j] = (x * w)[i,j] = \sum_{m=-M}^{M} \sum_{n=-N}^{N} x[i+m, j+n]\, w[m,n]$$

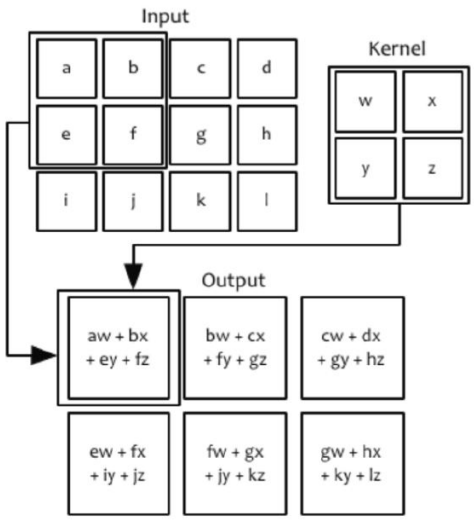


FIGURE 2.5: 2D Convolution [4].

The key characteristics of this kind of layer are:

- local connectivity;
- shared weights.

The **local connectivity** is a fundamental in the architecture of a CNN. In fact, when dealing with high-dimensional inputs such as images, it is infeasible to connect all the neurons to all neurons because of the computational cost [3]. Instead, CNN connect each neuron to only a local region of the input, i.e. the receptive field of the neuron which is equivalent to the filter size of the convolutional layer. Overall we can say that the connections are local in space along width and height, but always full along the entire depth of the input volume. Thanks to this type of architecture the learnt feature extractors produce a good response to a spatially local input patterns.

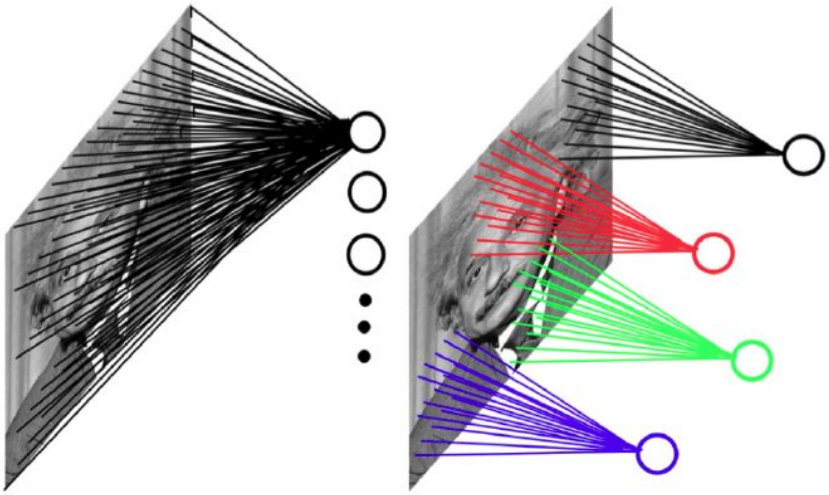


FIGURE 2.6: Fully connected NN vs Locally Connected NN [4].

One other important concept of the convolutional layers are the **shared weights** . In fact, weights are parameter that must be learned during the training phase and can be in a very large number. Therefore, weights sharing is used in convolutional layers to control the number of parameters and reducing computational costs. This scheme is based on a reasonable assumption [3]: if one feature is useful in the spatial position (x_1, y_1), then it should also be useful at the near position (x_2, y_2). This means that neurons are constrained to share the same weights with the nearest neurons. Replicating neurons in this way allows for *spatial invariance*, i.e. features can be detected regardless of their position in the input. Gradient descent is still used to learn such shared parameters, with only a small change to the original algorithm.
In Figure 2.7 (courtesy of [1]) is shown an example of the parameter sharing scheme.

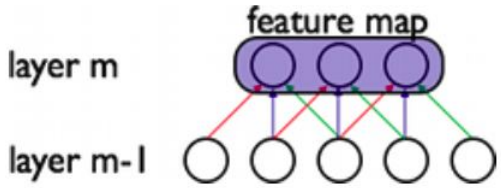


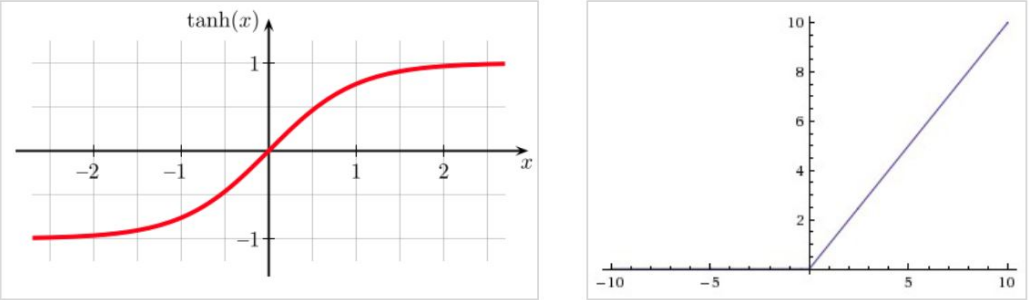
FIGURE 2.7: Three hidden neurons belonging to the same feature map share the weights of the same color which are constrained to be identical.

[1] http://deeplearning.net/tutorial/lenet.html

Non-Linear Layers

The non-linear layers (RELU) were introduced to increase the nonlinearity of the entire architecture without affecting the receptive fields of the convolution layers. A non-linear layer consists in a layer of neurons that applies a chosen non-linear function that can be:

- $f(x) = \max(0, x)$
- $f(x) = \tanh(x)$
- $f(x) = |\tanh(x)|$
- $f(x) = (1 + e^{-x})^{-1}$



$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \qquad f(x) = \max(0, x)$$

FIGURE 2.8: Examples of activation function for RELU layers: $tanh(x)$ and $max(x)$ [4].

Pooling Layers

Usually pooling layers (POOL) are periodically insert in-between successive convolutional layers in a CNN architecture. The task of the pooling layers is to progressively reduce the spatial size of the representation in order to reduce the amount of parameters to be computed and therefore the computational cost. Moreover, they are also useful to control overfitting. In Figure 2.9 we can see how the pooling layer drastically reduces the number of features.

In practice, this kind of layer divides the input image into a set of non-overlapping rectangles and, for each of the obtained sub-regions, it outputs the maximum or the average value of the features in that region as we can see in Figure 2.10. In fact, the user can chose between two main pooling operation:

- Max Pooling: outputs the maximum value of the feature within a rectangular neighborhood;
- Average Pooling: outputs the average value of the feature within a rectangular neighborhood which possibly is weighted by the distance from the central pixel;

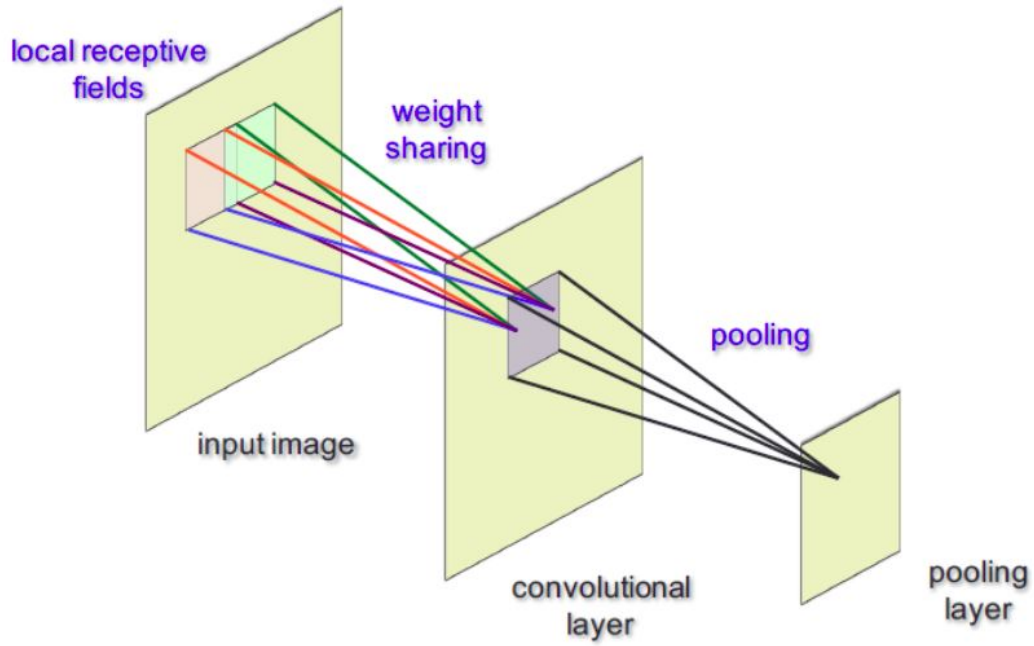


FIGURE 2.9: Sequence of the input, the convolutional and the pooling layers [4].

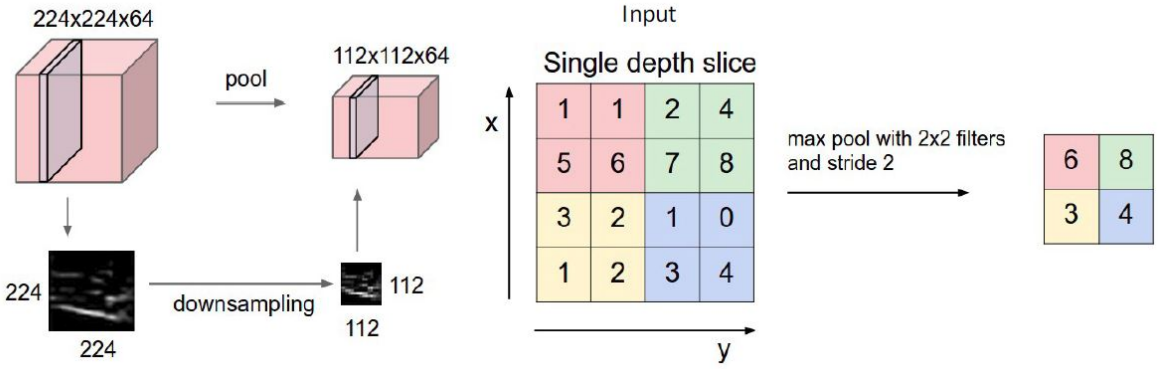


FIGURE 2.10: POOL downsamples the volume spatially, independently in each depth slice of the input volume so the volume depth is preserved. In this example, the input volume is pooled with a 2×2 max filter, i.e. each max is taken over 4 numbers [3].

Fully Connected Layers

The fully connected layer (FC) is the final layer of each CNN and it acts as simple classifier: given the features extracted in the previous layers it returns the classification output [3]. In this type of layers, neurons have full connections to all activations in the previous layer, as in classical NN. In some CNN the fully connected layers can be more than one.
In Figure 2.11 we can see a classical Deep Learning architecture which ends with a FC layer as classifier.

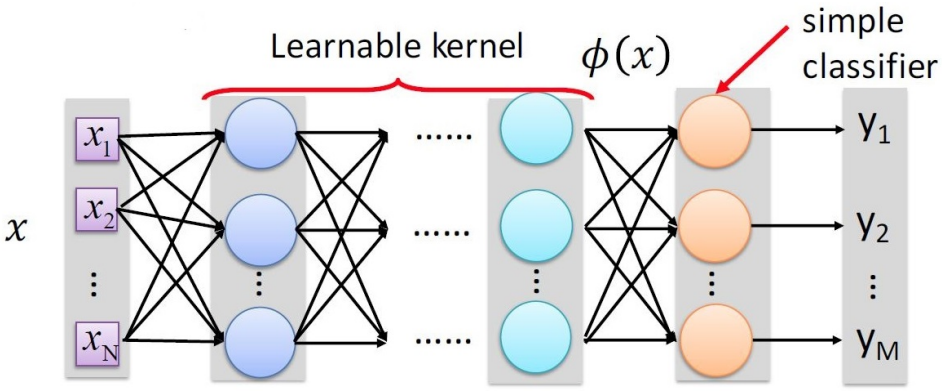


FIGURE 2.11: Deep Learning Architecture with the final Fully Connected Layer (image extracted from the Bioinformatic course slides).

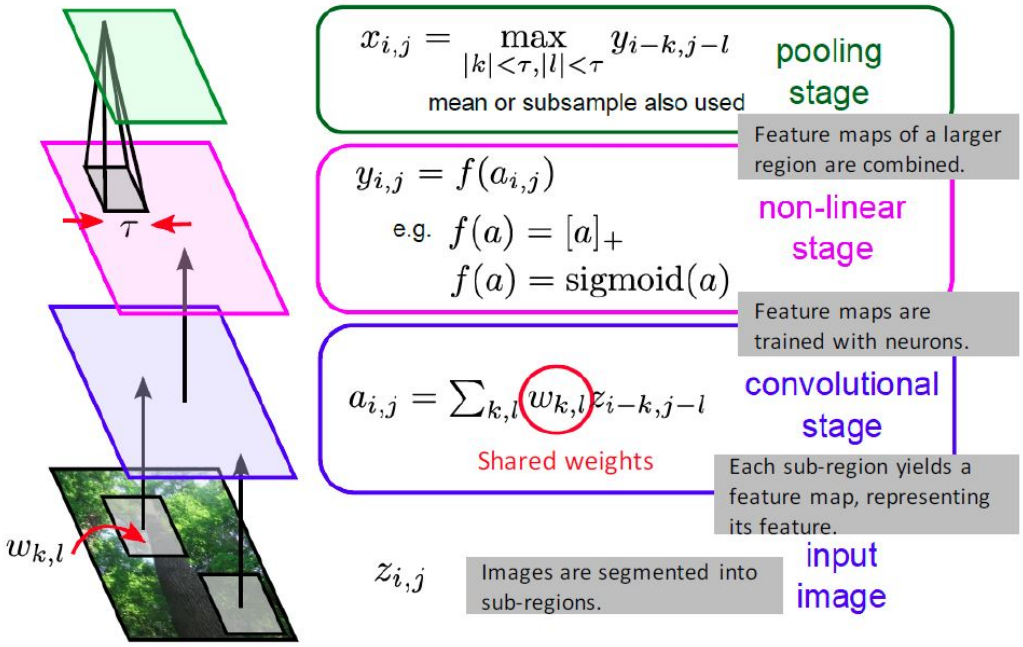


FIGURE 2.12: Building-blocks for CNN (image extracted from the Bioinformatic course slides).

2.2 Semantic Segmentation

Semantic segmentation can be considered one of the key problems to be solved in Computer Vision. It is an high-level task which is fundamental for a huge number of applications which require a complete scene understanding. These applications can be autonomous driving, human-machine interaction, biomedical images analysis, augmented reality and so on and can involve 2D images, video, and even 3D or volumetric data [6].
Such problem has been faced in the past using both traditional Image Processing techniques but also Machine Learning ones. In the last years Machine Learning, and in particular Deep Learning architectures like CNNs, outperformed traditional techniques by a large margin in terms of accuracy and sometimes even efficiency [6]. In particular Deep Learning overcame traditional Machine Learning methods thanks to its ability to learn also the appropriate feature representation for a specific problem in an end-to-end fashion instead of using hand-crafted features which require domain expertise and too much fine-tuning [6].
These new techniques are in a rapid and continuous evolution which make it difficult to find complete and updated reviews of the State of the Art. For this Section I referred to [6], where you can find a more deep description of the analyzed problems and methods.

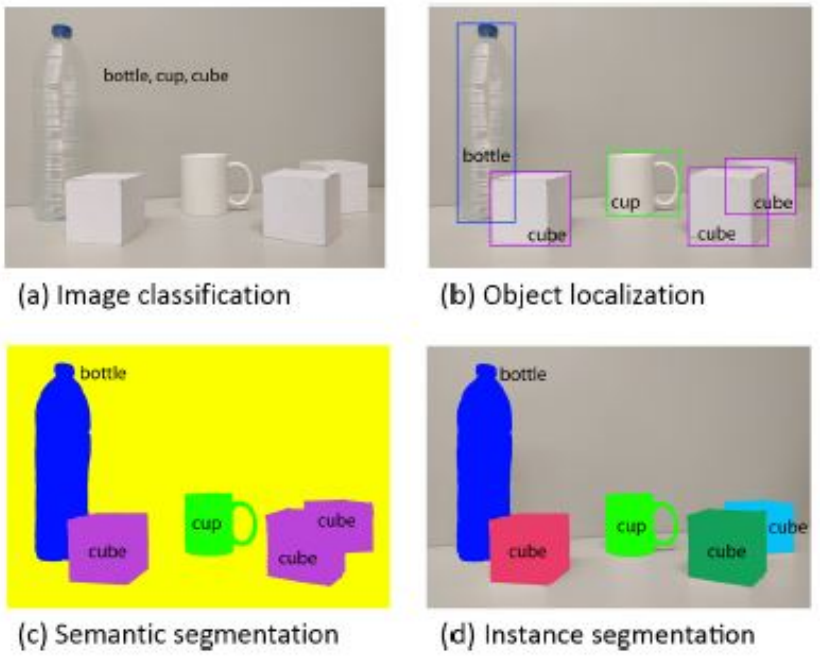


FIGURE 2.13: Evolution of scene understanding from coarse-grained to fine-grained inference: classification, detection or localization, semantic segmentation and instance segmentation [6].

In order to understand how semantic segmentation works and how Deep Learning architectures try to implement it, we have to recall the evolution of scene understanding from raw to the so called fine-grained inference [6]. This evolution is shown in Figure 2.13.
The first step can be considered *classification*, which consists of "making a prediction for a whole input, i.e. predicting which is the object in an image or even providing a ranked list if there are many of them" [6].

The second natural step consists of adding to the classes some additional information regarding the spatial location of those classes, e.g. centroids or bounding boxes [6]. This step is called *detection* or *localization*.
The third step is *semantic segmentation* which tries to make "dense predictions inferring labels for every pixel; this way, each pixel is labeled with the class of its enclosing object or region" [6]. Semantic segmentation can already be considered fine-grained inference but further improvements can be made. *Instance segmentation*, for example, tries to separate labels for different instances of the same class and *part-based segmentation* tries to perform a low-level decomposition of already segmented classes into their components [6].

2.2.1 Problem formulation:

The problem of semantic segmentation, i.e. assigning a label to each pixel, can be formally formulated defining the label space [6]:

$$L = \{l_1, l_2, ..., l_k\} \tag{2.1}$$

Each element of the label space represents a different class, e.g. cat, dog, car, healthy, non-healthy, background and so on. This label space has k elements, i.e. k possible states which are usually extended to $k + 1$ if l_0 is treated as background or a void class [6]. In the same way we can define a set X of random variables [6]:

$$X = \{x_1, x_2, ..., x_N\} \tag{2.2}$$

Usually, X can be a $2D$ image of $W \times H = N$ pixels x but this formulation allows the extension to any dimensionality such as volumetric data or hyperspectral images [6]. Given these assumptions, the pixel labeling problem can be reduced to the following formulation: "find a way to assign a state from the label space L to each one of the elements of X" [6].

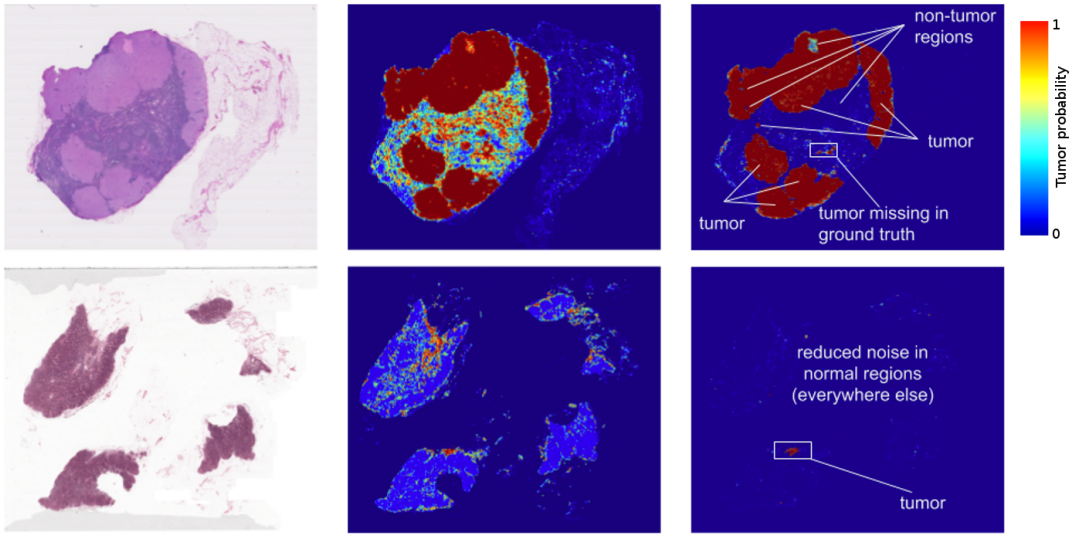


FIGURE 2.14: Left: Images from two lymph node biopsies. Middle: earlier results of a Deep Learning tumor detection. Right: final heatmap of the tumor probability [7].

2.3 CNNs for Semantic Segmentation

Nowadays, the most successful State of the Art Deep Learning techniques for semantic segmentation are based on famous CNNs architectures such as AlexNet, VGG-16, GoogLeNet and ResNet [6]. Obviously these networks are opportunely modified in order to have a pixel classification instead of a whole-input classification.
In nutshell we can say that a general semantic segmentation architecture can be seen as an encoder network followed by a decoder network [8]. The encoder is usually a pre-trained classification network like VGG/ResNet or another one of those cited before, while the decoder is mostly where these architectures differ.
In the encoder part features with an increasing level of abstraction are learnt, i.e. features related to the context. But semantic segmentation requires a pixel-level classification, therefore it is necessary a mechanism to project the features learnt at different stages of the encoder onto the pixel space in order to get a dense classification and no more a simple classification of the whole input. This need is implemented in different architectures using different mechanisms (skip connections, pyramid pooling etc) as a part of the decoder network. For a more deep explanation see [6] and [8].

2.3.1 Patch Classification

Patch classification can be considered the first approach to semantic segmentation using CNNs [9] [10]. It was most used because CNNs usually present fully connected layers which make it necessary for fixed size input images. The dimensions of the patches were simply chosen according to this size.
There are lots of different architectures (for example the famous one of Ciresan et al. [11]) but overall we can say that each pixel is separately classified into classes using a patch of image around it [10]. These patches can be overlapped or not and can be obtained using sliding windows. At the end, all the information are put together in order to get the final segmentation map.

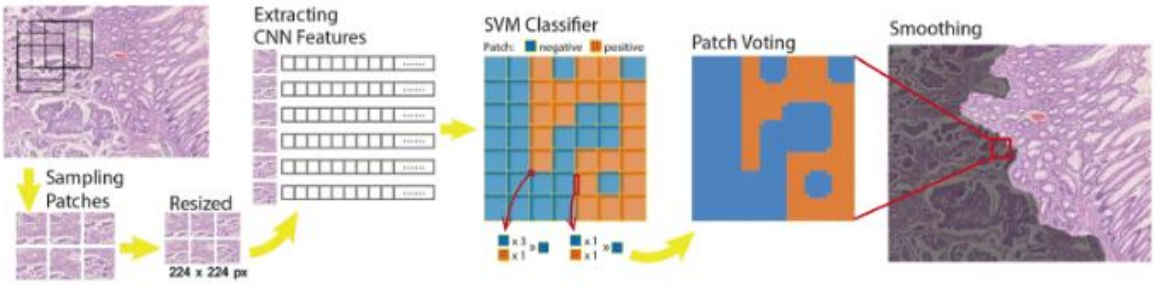


FIGURE 2.15: Example of images segmentation using patch classification: single patches are classified using a CNN. The results are put together in a tumor probability heatmap and then an SVM classifier evaluates if the tumor is actually present or not [12].

This strategy has two main drawbacks [10]. First of all there is a redundancy due to the fact that the network must be run separately for each patch. This problem is emphasized when the patches overlap and makes the network very slow. Second but most important is the right tuning of the dimensions of the patches in order to find a trade-off between localization accuracy and the use of context. Larger patches require more max-pooling layers with a consequent reduction of the localization accuracy, while small patches reduce the field of view, i.e. allow the network to see only little context [10].

2.3.2 Fully Convolutional Network

In 2015, a team of the Berkley university published a work that upset the world of semantic segmentation [6]. Long et al. designed the Fully Convolutional Network (FCN), a CNN architecture for dense predictions without any fully connected layers [13]. It was the first work that showed how CNNs can be trained end-to-end for semantic segmentation tasks. Currently, almost all the successful State of the Art Deep Learning techniques for semantic segmentation stem from this model [6].
Their idea was simple: modifying existing and successful classification models such as AlexNet, VGG16, GoogLeNet, and ResNet into fully convolutional ones by replacing the fully connected layers with convolutional ones [6]. The absence of fully connected layers allows to output spatial maps instead of classification scores and to use input images of any size.
The second advantage is the fastness of this method with respect to the patch classification approach. Moreover, it defeated traditional methods accuracies on standard datasets, while preserving its efficiency [6].

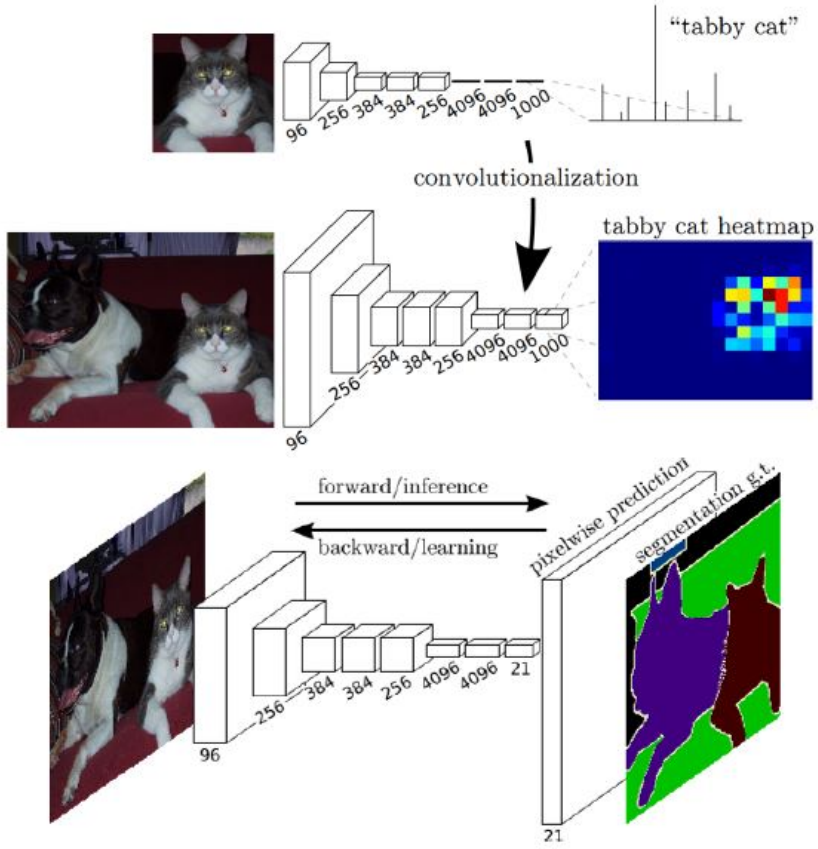


FIGURE 2.16: Transformation of a classical CNN to FCN by replacing fully connected layers with convolutional ones and adding deconvolution layers for up-sampling [6].

FCN uses existing CNNs as encoder networks to learn a hierarchy of features [6]. With respect to the classical AlexNet, VGG16, GoogLeNet, and ResNet, after each convolution block, pooling layers were introduced. This operation enabled the succeeding block to extract more abstract, class-salient features, i.e. capture the context and increase the field of view [8]. But we have to remember that pooling is a sampling operation which loses the spatial information fundamental for semantic segmentation.

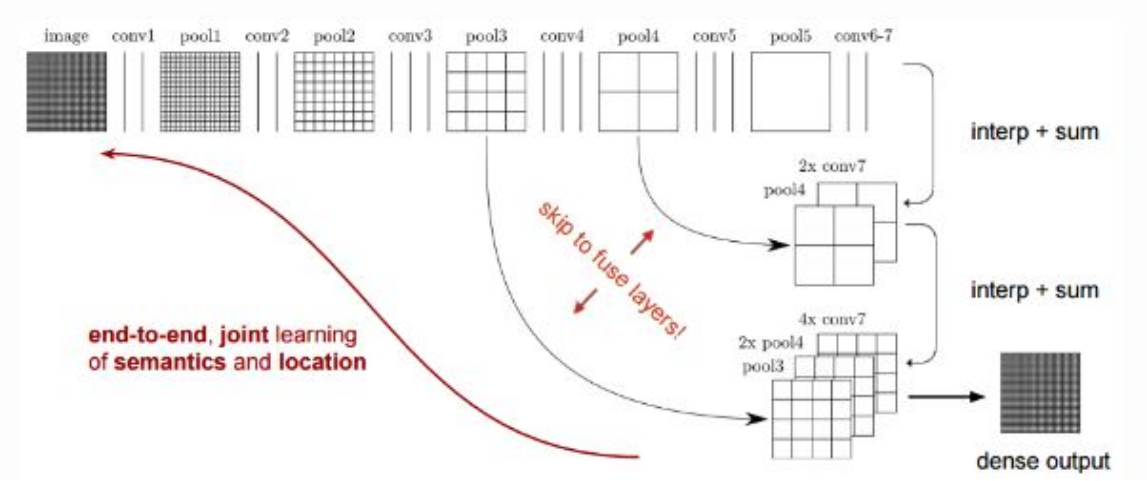


FIGURE 2.17: The FCN-32s Architecture [8]

In any case the encoder part does not produce a dense per-pixel classification. Therefore, those features maps are upsampled through an operation called deconvolution, i.e. a fractionally strided convolution.
To overcome the problem of the loss of spatial information, two different classes of architectures evolved in literature [9]:

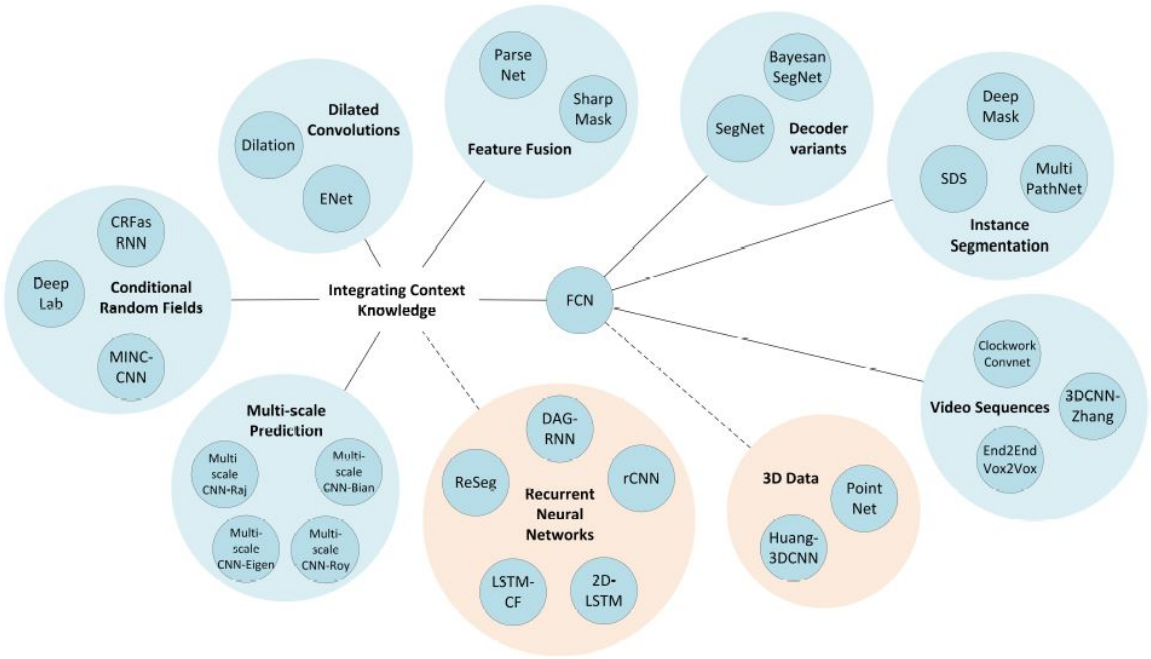


FIGURE 2.18: Visualization of the developed methods starting from FCN.[6].

- Encoder-decoder architectures: the encoder part gradually reduces the spatial dimension with pooling layers while the decoder one gradually recovers the object details and spatial dimension. There are usually shortcut connections from encoder to decoder to help the decoder network to recover the object details better.

- Dilated/atrous convolutions architectures: use of particular pooling layers which enable to recover the spatial information and object details.

2.3.3 Encoder/Decoder Architectures

In this subsectiontwo encoder/decoder architectures of particular interests are presented but in literature there are lots of implementations of such paradigm like ENet [14], LinkNet [15], G-FRNet [16], RefineNet [17], Large Kernel Matters [18] and Fully Convolutional DenseNet (also known as Tiramisu) [19].

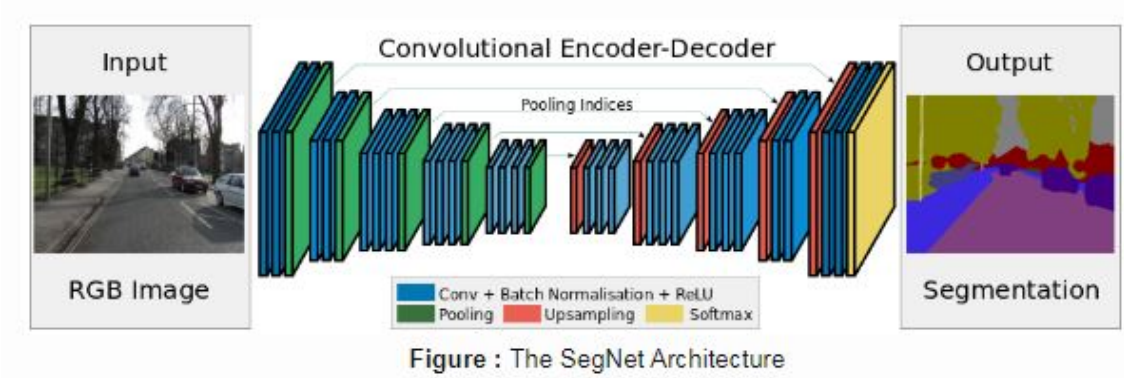


FIGURE 2.19: The SegNet Architecture [8].

SegNet

SegNet is a network that belongs to the encoder/decoder architectures. While the encoder part is a classical one, the decoder part is the novelty of this network. In fact, it consists of a set of up-sampling and convolution layers followed by a final softmax classifier which can predict pixel-wise labels for an output which has the same resolution as the input image.

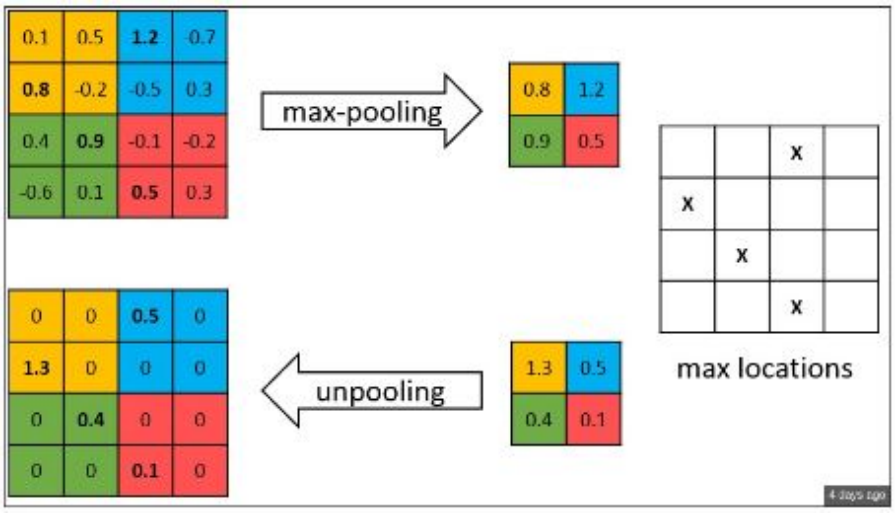


FIGURE 2.20: SegNet: the indices at each max-pooling layer in encoder are stored and later used to upsample the corresponding feature map in the decoder by unpooling it using those stored indices.

Each up-sampling layer in the decoder part corresponds to a max-pooling one in the encoder part. In particular, each up-sampling layer uses pooling index computed in the corresponding max-pooling layer to perform non-linear up-sampling as it shown in Figure 2.20. This eliminates the need for learning the parameters of the up-sampling while FCN-based networks use learnable deconvolution filters to upsample. Thanks to this trick the high-frequency information is preserved but the neighborhood information is lost [8].
The up-sampled maps are sparse and are then convolved with trainable filters to produce dense feature maps. After that, the upsampled feature maps are added element-wise to the corresponding feature map generated by the convolution layer in the encoder part. When the feature maps have been restored to the original resolution, they are fed to the softmax classifier to produce the final segmentation.
All the information were taken from this review [6] and from the original paper [20].

UNet

UNet is the second encoder/decoder architecture I chose to review because it was born for biomedical segmentation purpose. This network has many successful biomedical applications like, for example, the segmentation of neuronal structures in EM stacks (winner of ISBI 2012), where it defeated the network of Ciresan et al. [11] and the cell segmentation in light microscopy images (winner of ISBI cell tracking challenge 2015). This network is expressly inspired from the FCN described in Section 2.3.2 and was modified in order to work with very few training images and yield precise segmentations.

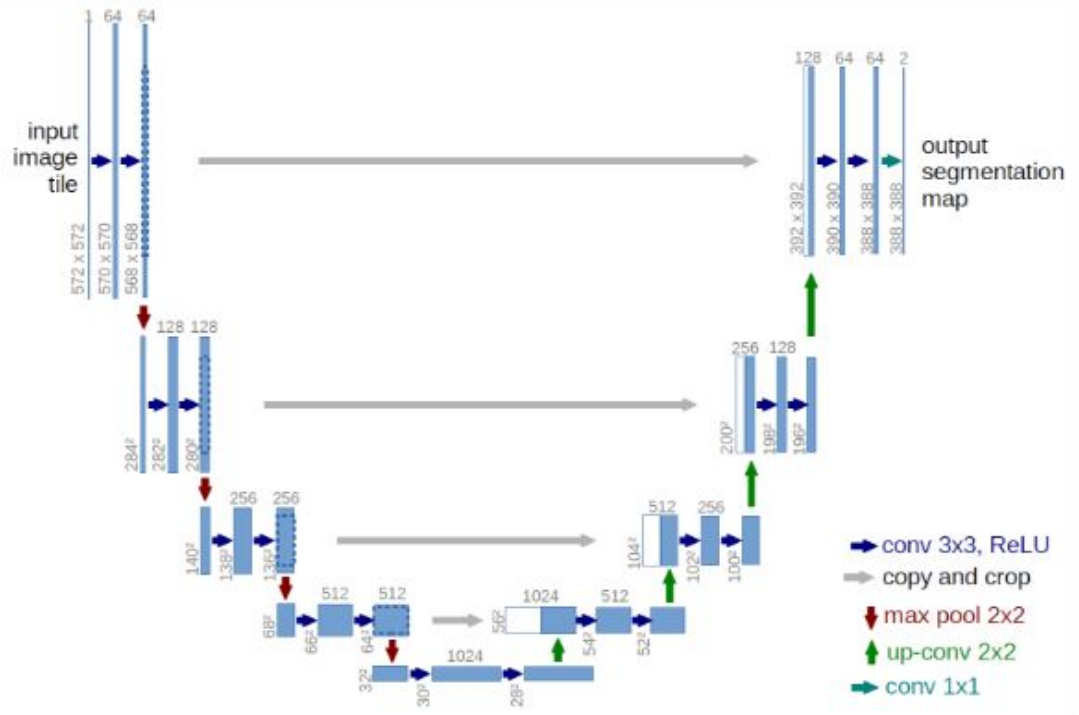


FIGURE 2.21: Example for an UNet with 32 × 32 pixels in the lowest resolution. Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box, the x-y-size at the lower left edge of the box. White boxes represent copied feature maps [10].

In the original paper [10], the authors describe the encoder part of the network as a contracting path and the decoder part as an expansive path as it is shown in Figure 2.21. These two path are more or less symmetric and can be drown graphically using an U-shape from which derives the name of the network.
The contracting path can be considered a classical CNNs architecture made by convolutional, non-linear (RELU) and max-pooling layers. In particular it consists of 5 blocks, each one containing two 3×3 convolutions blocks (unpadded convolutions), each followed by a rectied linear unit (RELU) and a 2×2 max pooling operation with stride 2 for down-sampling (each down-sampling step doubles the number of feature channels).
Each block in the expansive path is made by an up-sampling of the feature map followed by a 2×2 "up-convolution" that split in half the number of feature channels, a concatenation with the correspondingly cropped feature map from the contracting path, and two 3×3 convolutions, each followed by a RELU. The cropping is necessary due to the loss of border pixels in every convolution. In this way high resolution features from the contracting path are combined with the upsampled output in order to sum up all the necessary information for the segmentation.
The last layer is a 1×1 convolution used to map each 64-component feature vector to the desired number of classes. In total the network has 23 convolutional layers.
The important modification with respect to FCN is that in the up-sampling part there is a large number of feature channels, which yield to the propagation of context information to the pixel-level.

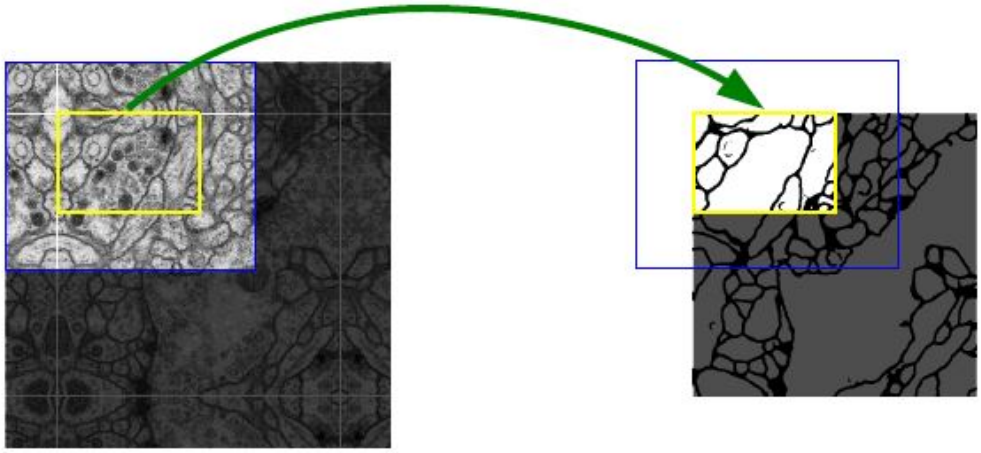


FIGURE 2.22: Overlap-tile strategy for seamless segmentation of arbitrary large images (here segmentation of neuronal structures in EM stacks). Prediction of the segmentation in the yellow area, requires image data within the blue area as input. Missing input data is extrapolated by mirroring[10].

Due to the fact that U-Net has no FC layers, arbitrarily large images can be segmented through an overlap-tile strategy. Moreover, U-Net classifies only pixel for which the full context is available while for the border pixels the missing context is recovered by a mirror strategy. Therefore, the input tile dimension dimensions must be chosen in such a way that all the 2×2 max-pooling layers get an input with even dimensions allowing a tiling without breakpoints like in Figure 2.22.

2.4 Enhancing Techniques

The major problem when dealing with Deep Learning and, more in general, Machine Learning models, is the lack of large and really representative datasets. In fact, the larger is the dataset the better are the results in terms of accuracy and generalization capabilities.
Most of the times, it is really hard to obtain enough and good data, especially due to the economical costs, the time required for the collection and so on. These problems are amplified in the biomedical field where the acquisition of data is really expensive, time-consuming, strictly dependent on the standard of acquisition and storing. Moreover, these data come from humans and this implies an huge number of ethics problems like the one connected to the person privacy. In addition, one of the main problem is the building of the ground truth for feeding supervised learning models. For this purpose it is required the expertise of one or more specialist who are able to classify data by hand. This is a really time-consuming and expensive operation which suffers also of the subjectivity of the specialist.
These are the main reasons why large and representative datasets are still difficult to find in the biomedical field. Most of them are made available for challenges or calls in international conference like the hystopathological datasets summarized in section 3.2.2.
Therefore, while waiting for larger and standardized biomedical datasets, we can exploit some tricks to overcome this problem. Two of them are the so-called *Transfer Learning* and *Data Augmentation*

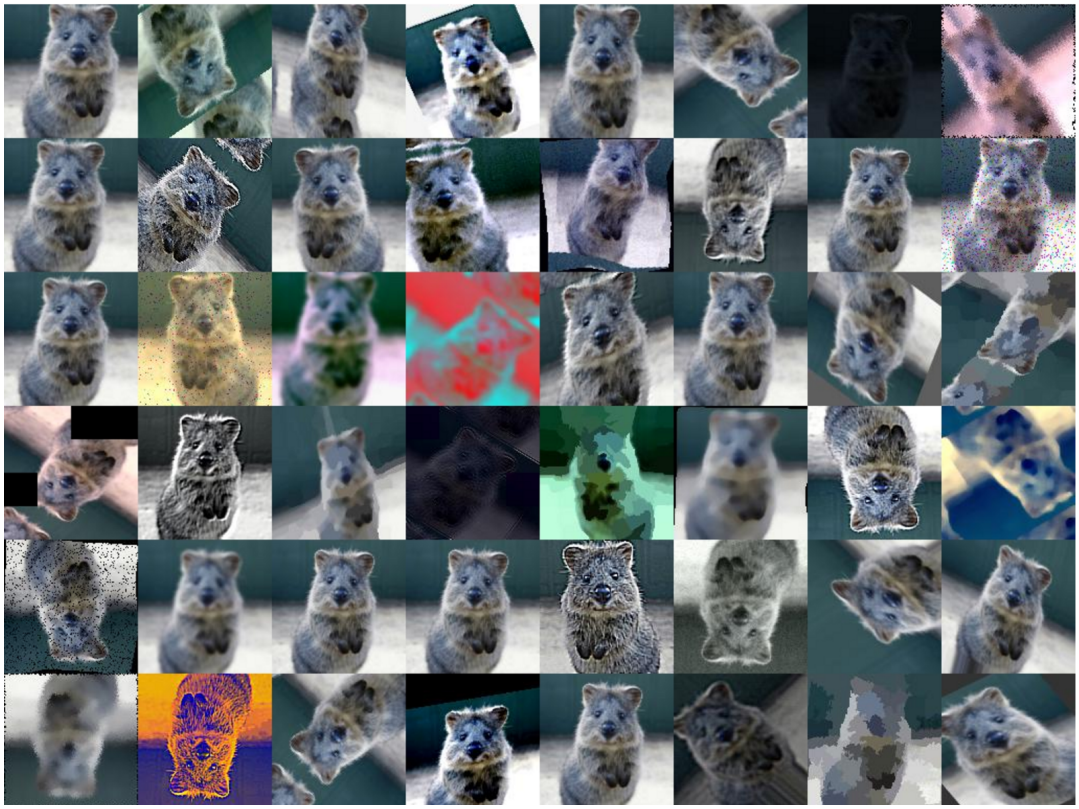


FIGURE 2.23: Data augmentation example [21]

2.4.1 Data Augmentation

Data augmentation is a really common way to increase the number of training samples in order to improve the performances of Machine Learning models. Moreover, it seems also to benefit models in terms of time of convergence, generalization capabilities, preventing overfitting, regularizing the model, balance the classes, and even synthetically produce new samples that are more representative for the use case or task at hand [6].
Data augmentation is obtained by applying a set of transformations in either data or feature spaces, or even both [6]. For example, in the case of CNNs for semantic segmentation the transformation are performed in the data space [10] in order to generate new samples to feed the deep model. The transformation that can be applied in such case can range from space transformation like translation, rotation, warping, vertical or horizontal mirroring, scaling, color space shifts, changing the brightness, pads, crops, etc. Obviously, these transformation must be traced also in the segmentation masks, while in the case of classification tasks this would not be a problem.
The Python library Keras allows to do data augmentation on the fly, i.e. during the training without saving the new images. In this case the training phase is a longer but it solves lots of memory problems. In the field of biomedical segmentation images, data augmentation can be useful when few training data are available. Moreover, the most common variation in real tissues can be simulated through data augmentation like for example the deformation of tissues, the change in the brightness and so on in order to led the network invariant to such deformations. For example, the designer of UNet in [10] applied elastic deformation to their small dataset improving relevantly their performances.

2.4.2 Transfer Learning and Fine Tuning

A second way that allows to overcome the lack of data is Transfer Learning, i.e. "extract the knowledge from one or more source tasks and apply the learned knowledge to a novel target task" [22]. Transfer Learning was first inspired by humans who can exploit knowledge learned during their life to solve new problems faster and better than if they did not own this prior knowledge [22].

The main difference between the traditional machine learning process and the transfer learning one is shown in Figure 2.24: the first one aims to learn each task from scratch, while the second one aims to transfer the knowledge acquired in a previous task to a novel one. The second solution is particularly useful when the data available for the novel task are less or weakly annotated.

Moreover, when dealing with deep learning models, training a network from scratch may be really difficult not only for the lack of enough data that leads to bad performances but also for the required computational time and resources [6]. Reaching the convergence can take really long time and most of the time GPUs are required. In this situation Transfer Learning can play a key role by reducing drastically the required computational time and resources.

Transfer learning is a really general denomination which can be use to identify many types of knowledge transfer but in this work I will focus particularly on the so-called *Fine Tuning*. This methods consists in literally fine-tuning the weights obtained by a pretrained network by continuing the backpropagation [2].

[2] http://cs231n.github.io/transfer-learning/

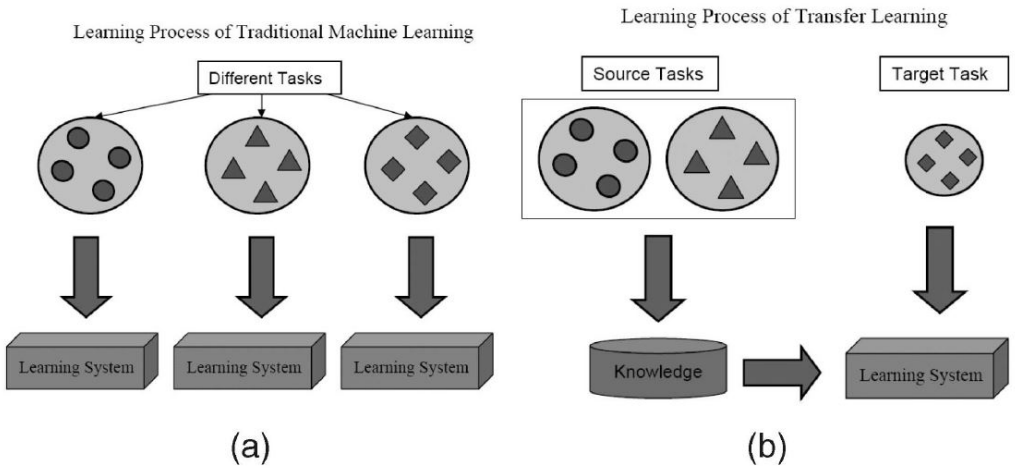


FIGURE 2.24: The main difference between (a) traditional machine learning and (b) transfer learning lies in the learning processes [22].

Fine-Tuning can be applied to the entire network or only to some layers. This means that is possible to keep some layers fixed to their previous weight while the others can be fine-tuned. Usually the fixed layers are the earliest ones because they extract really general features while the closer to the output the more abstract and specific for the task are the features [23]. So we can say that higher layers are too much chained to the original dataset and task.

It has been shown that starting from pretrained weights instead of randomly initialized ones lead to better performances, even if the available dataset is large enough and the required computational time and resources are feasible [6]. The work of [23] focused on such idea: transferring features is better than a random initialization also if the two task are really different. Obviously the more different the two tasks, the less is the transferability [6].

However, when choosing the transfer learning strategy, some considerations taken into account. The two most important ones are the size of the new dataset and its similarity to the original one [3]:
If the new dataset is small and similar to the original dataset the fine-tuning strategy could lead to overfitting because higher level layers of the CNN may be relevant for the new dataset as well. In this case is better to start only retraining the last classifier. On the contrary, if the new dataset is similar to the old one but enough large, the overfitting danger decreases and fine-tuning can be used.
If the new dataset is small and different from the original one, the fine-tuning must be performed on more layers because the learnt features are too much specific for the old dataset. Otherwise, if the new dataset is large and very different from the original one the network can be retrain from scratch. However, as said before, initialize with weights from a pretrained model would improve the performances.

[3] http://cs231n.github.io/transfer-learning/

Chapter 3

State of the Art

3.1 Digital Pathology and Microscopy

Pathology is one of the major field of modern medicine and it consists in the analysis of tissues and cells samples for diagnostic purposes [24]. Traditionally, the pathologist bases his analysis only on experience and the examination is done manually with the aid of a microscope. These kind of processes can suffer from considerable *inter* and *intra-reader variability* and many others problems. The digitization of pathology aims to solve many of these criticisms, but there are still lots of open problems related especially to the amount of data required for building new algorithms and the computational load when dealing with high resolution images. An additional issue for the development of automated methods is the large variability in the analyzed data which is given by factors like the cell types, the cell densities, the stains, the magnification levels, and so on.

FIGURE 3.1: Number of patients per pathologist around the world [25].

3.2 CNNs Applications in Digital Pathology and Microscopy

The CNNs applications that deal with digital pathology and microscopy focus on three main aspects [26]:

- detection, segmentation and classification of nuclei;
- segmentation of large organs;
- detection and classification of diseases and lesions;

All these works were possible thanks to the increasing availability and the high resolution of digital pathological images. In fact, such as for general machine learning algorithms, deep learning requires very large datasets to train their models. In fact, the larger the dataset is, the more performing is the model.

In the tables below a schematic overview of papers involving CNNs in histological applications is reported. These data have been extracted from a complete review on deep learning in medical image analysis [26]. Moreover they have been integrated with more specific informations such as performances and dataset descriptions extracted directly from every paper.

These three tables follow the three main research branches in this field and in each table the papers are ranked by the publication year. Each work is summarized using the following attributes and abbreviation [26]:

- the reference to the paper in the bibliography;
- the specific research topic;
- the **staining and imaging techniques** (SIM):

H&E: Hematoxylin and eosin staining,
TIL: Tumor-infiltrating lymphocytes,
IHC: Immunohistochemistry,
RM: Romanowsky,
EM: Electron microscopy,
PC: Phase contrast,
FL: Fluorescent,
IFL: Immunofluorescent,
TPM: Two-photon microscopy,
CM: Confocal microscopy,
Pap: Papanicolaou.;

- the characteristics of the dataset:

GT: ground truth,
OP: open,
MS: microscope,
M: magnification;

- a brief description of the method;

- the performances:

  P: precision,
  R: recall,
  TNR: specificity,
  ACC: accuracy,
  F: F-score,
  F1: F1-score,
  DSC: dice similarity coefficient,
  JI: Jaccard indices,
  AVD: Average Hausdorff Distance.

Other abbreviations are: BCC: Basal cell carcinoma.

TABLE 3.1: Nucleus detection, segmentation, and classification [26]

Ref	Topic	SIM	Method	Dataset	Performance
[27]	Mitosis detection	H&E	CNN-based pixel classifier	MITOS	P=0.88, R=0.70, F1=0.78
[28]	Detection of basal cell carcinoma	H&E	Convolutional auto-encoder NN	BCC	ACC=0.921, P=0.901, R=0.887, TNR=0.941, F=0.894
[29]	Mitosis detection	H&E	Combines shape based features with CNN	ICPR 2012	F1=0.659 (color scanners), F1=0.589 (multispectral)
[30]	Mitosis detection	H&E	Cascaded ensemble of CNN and handcrafted features	ICPR 2012	F=0.7345
[31]	Cell segmentation	EM	U-Net with deformation augmentation	ISBI 2012-2014-2015	WarpingError=0.000353, RandError=0.0382, PixelError=0.0611
[32]	Mitosis detection	Live-imaging	CNN-based patch classifier	NIH3T3 scratch assay cultures	AUC=0.91, F=0.89
[33]	Segmentation of cytoplasm and nuclei	H&E	Multi-scale CNN and graph-partitioning-based method	Self-made [a]	PPV=0.94, NPV=0.92, F1=0.912 (all nuclei evaluation)
[34]	Nucleus detection	Ki-67	CNN model that learns the voting offset vectors and voting confidence	44 Ki67-stained NET microscopy images [b]	P=0.852, R=0.794, F1=0.8152 (for DV-1)
[35]	Nucleus detection	H&E, Ki-67	CNN-based structured regression model	TCGA [c]	P=0.919, R=0.909, F1=0.913
[36]	Cell segmentation	FL, PC, H&E	fCNN for cell bounding box proposal and CNN for segmentation	ISBI [d]	Average P=0.963, final R=0.996
[37]	Mitosis detection	H&E	"Crowd sourcing" layer into the CNN framework	MICCAI-AMIDA13 [e]	AUC=0.8695, F1=0.6133
[38]	Nucleus classification	IHC	CNN-based patch classifier	Self-made [f]	Benign vs malignant cells: P=0.82, R=0.80, F1=0.81, Prostate vs renal cells: P=0.80, R=0.80, F1=0.80
[39]	Nucleus classification	IFL	Classification of Hep2-cells with CNN	ICPR 2014	Avarage ACC= 0.748
[40]	Nucleus classification	IFL	Classification of Hep2-cells with CNN	ICIP 2013 HEp-2	
[41]	Nucleus detection	H&E	Combination of CNN and hand-crafted features	Self-made [g]	P=0.772, R=0.725, F=0.748
[42]	Mitosis detection	PC	Hierarchical CNNs for patch sequence classification	Self-made [h]	P=0.991, R=0.972, F=0.986
[43]	Classification of mitochondria	EM	CNN-based patch classifier	Self-made [i]	ACC=0.942, R=0.985, TNR=0.915
[44]	Nucleus classification	FL	Classification of Hep2-cells using transfer learning (pre-trained CNN)	ICPR 2012	ACC=0.771, ACC=0.915 (Leave-one-out)
[45]	Tubule nuclei detection	H&E	CNN-based classification of pre-selected candidate nuclei	Self-made [j]	AUC=0.76
[46]	Nucleus detection and classification	H&E	CNN with spatially constrained regression	Self-made [k]	AUC=0.917, F1=0.784
[47]	TIL detection	H&E	CNN-based classification of superpixels	Self-made [l]	AUC=0.99, F=0.94
[**veta2016cutting**]	Nuclear area measurement	H&E	A CNN directly measures nucleus area without requiring segmentation	Self-made [m]	
[48]	Subtype cell detection	H&E	Combination of two CNNs for joint cell detection and classification	TCGA [n]	
[**Xie2016**]	Nucleus detection and cell counting	FL and H&E	Microscopy cell counting with fully convolutional regression networks		P=0.8029, R=0.8683, F1=0.8215
[49]	Nucleus segmentation	H&E, IHC	CNN and selection-based sparse shape model	Brain tumor,Neuroendocrin tumor and Breast cancer images, GT by two pathologists	Brain: P=0.72, R=0.88, F1=0.77; NET: P=0.84, R=0.93, F1=0.88; Breast: P=0.71, R=0.88, F1=0.78
[50]	Nucleus detection	H&E	Stacked sparse auto-encoders (SSAE)	Self-made [o]	P=88.84, R=82.85, F=84.49, Average P=78.83
[51]	Nucleus detection	Various	DCNN to detect cells in whole-slide images		P=0.83, R=0.84, F1=0.83
[52]	Glial cell segmentation	TPM	fCNN with an iterative k-terminal cut algorithm	Self-made [p]	average F1 = 0.894
[53]	Classification of leukocytes	RM	CNN-based patch classifier		
[54]	Nucleus classification	H&E	Classifies cellular tissue into tumor, lymphocyte, and stromal	TGCA and lung cancer dataset (NLST)	
[55]	Cell segmentation	H&E	Multi-scale CNN	ISBI 2015 [q]	ISB dataset: P=0.95, R=0.93, F1=0.94 SZU dataset: P=0.94, R=0.92, F1=0.91

[a] 53 slides with normal and abnormal nuclei collected locally, M:40x, Olympus BX43 MS, manual GT by pathologists

[b] 400×400 pixels

[c] 32 images of breast cancer cells from The Cancer Genome Atlas (TCGA), 400×400 pixel, M:40×

[d] Fluo-N2DL-HeLa dataset from ISBI cell tracking challenge: 2 time-lapse sequences (92 frames each) of fluorescent HeLa cells cultured and imaged on two dimensional surface. The ground truth (GT) contains markers for all cells in all frames and segmentation masks for all cells in 2 frames from each sequence)

[e] MICCAI-AMIDA13 challenge dataset: OP, 23 patients, RGB images of suspect breast tissues, Aperio Scan Scope XT MS, M:40x, spatial resolution 0.25 microm/pixel, GT of two expert pathologists

[f] 8 ccRCC TMA images with 1272 labeled nuclei by two pathologists (890 benign and 382 malignant), OP.

[g] 15 images of colorectal cancer tissues (M:20x)

[h] 5 phase contrast video sequences each on with 1436 images ($1392x1040$ pixels) containing mitosis cells. GT: location and time of mitosis events in the video sequences. Data expansion to generate more positive training data

[i] 403 images 163 of healthy and 240 of cancer mitochondria from rat liver and cultured cells), EM 10 CR transmission electron microscope (Zeiss, Germany), size:1032×1376 pixels, $M : 20000x$

[j] WSIs extracted from 174 patients with positive to Breast Cancer, GT by a pathologist

[k] 100 images of colorectal adenocarcinomas from 10 WSIs from 9 patients, size:$500x500$ pixels, M:$20x$, Omnyx VL120 MS, GT by two pathologists (29756 nuclei were marked in 4 classes: epithelial, fibroblast, inflammatory and miscellaneous

[l] Breast tumor samples collected locally from 20 patients, 5 types of tumor and 3 histological grades). WSIs with M:$20x$, resolution:0.22 μm/pixel, average size:8.5×109 pixels, MS: Pannoramic 250 FLASH, GT by two pathologists

[m] breast cancer histopathology images locally with manually segmented nuclei by a pathologist (39 slides, M:$40x$, spatial resolution of $0.25\mu m$/pixel, size:4000×4000 pixels

[n] From the Cancer Genome Atlas 300 512×512 pixels lung cancer images, GT by pathologist, 3 classes: lymphocytes, stromal cells and tumor cells

[o] 537 breast cancer histological images collected locally, size:2200x2200 pixels, Aperio ScanScope digitizer, M:40, OP

[p] 3D images of mouse brains using two-photon MS, size:$640 \times 640 \times 25$ voxels, voxel resolution:$1 \times 1 \times 2\mu$m)

[q] (8 cervical cytology images containing 20-60 cells with 1024×1024 pixels) and Shenzhen University (SZU) Dataset (collected in the local hospital, 21 cervical cytology images with 1360×1024 pixels containing about 30-80 cells, GT by pathologists)

TABLE 3.2: Large organ segmentation [26]

Reference	Topic	SIM	Method	Dataset	Performance
[11]	Segmentation of neuronal membranes	EM	Ensemble of several CNNs with different architectures	ISBI 2012	WarpingError=0.000434 RandError=0.048 PixelError=0.06
[56]	Segmentation of colon glands	H&E	Used two CNNs to segment glands and their separating structures	Warwick-QU	train: P=0.91, R=0.85, F1=0.88 test: P=0.67, R=0.77, F1=0.68
[57]	Detection of lobular structures in breast	IHC	Combined the outputs of a CNN and a texture classification system	Self-made [a]	Highest F1 obtained by MTD (0.59) MDL (0.60) for the test set
[58]	Segmentation of colon glands	H&E	fCNN with a loss accounting for smoothness and object interactions	Warwick-QU	FCN+Smoothness: PixelACC=0.86, DSC=0.78, Inference=28.62s; FCN+Smoothness+Topology: PixelACC=0.76, DSC=0.80, Inference=28.63s
[59]	Segmentation of colon glands	H&E	A multi-loss fCNN to perform both segmentation and classification	Warwick-QU	classACC=0.89, pixelACC=0.92; Benign DSC=0.90, Malignant DSC=0.76
[60]	Neuronal membrane and fungus segmentation	EM	Combination of bi-directional LSTM-RNNs and kU-Nets	ISBI	PixelError=0.0215
[61]	Segmentation of neuronal structures	EM	fCNN with skip connections	ISBI 2012	specific metrics
[62]	Segmentation of colon glands	H&E	Compares CNN wit SVM using hand-crafted features	Warwick-QU	Hand-crafted+Alexnet: JI=0.77, DSC=0.87; Hand-crafted+Alexnet+Googlenet: JI=0.77, DSC=0.87
[63]	Volumetric vascular segmentation	FL	Hybrid 2D-3D CNN architecture	Self-made[b]	AVD=0.49, AUC=0.93
[**wang2016dee**]	Segmentation of messy and muscle regions	H&E	Conditional random field jointly trained with an fCNN	Self-made[c]	pixel ACC=0.90
[64]	Segmentation of colon glands	H&E	Used three CNNs to predict gland and contour pixels	MICCAI 2015 Gland Segmentation Contest	F1=0.771, DSC=0.815
[65]	Segmenting epithelium and stroma	H&E, IHC	CNNs applied to over-segmented image regions (superpixels)	D1: Breast cancer TMAs images D2: 27 TMAs of colorectal cancer [d]	ACC=1, F1=1, R=1, TNR=1, PPV=1, NPV=1
[66]	Segmentation of colon glands	H&E	Deep contour-aware CNN	Warwick-QU	Version1: F1=0.7692, DSC=0.8001 Version2: F1=0.9116, DSC=0.8974

[a] WSIs collected locally, MS:Aperio AT2, M:40x, resolution:0.253 μm/pixel, GT by one pathologist

[b] Vessel dataset acquired from mouse cortex and GFP-labelled human squamous cell carcinoma tumors using the FV1000 MPE two-photon laser scanning MS (Olympus)

[c] histology tissue collected in the local hosiptal. 200 images size:1000 $\times$ 1000 pixels, M:40x and resized to 10x, GT manual

[d] size:1128 $\times$ 720 pixels, M:20$\times$; Both manually labeled as EP or ST by expert pathologists

TABLE 3.3: Detection and classification of disease and other pathology applications [26]

Reference	Topic	SIM	Method	Dataset	Performance
[67]	Detection of invasive ductal carcinoma	H&E	CNN-based patch classifier	Self-made [a] GT by a pathologist	F=0.718, Balanced ACC=0.842
[68]	Patch-level classification of colon cancer	H&E	Multiple instance learning framework with CNN features	Sefl-made [b]	ACC=0.9781
[69]	Grading glioma	H&E	Ensemble of CNNs	TCGA [c]	GBM VS LGG: ACC=0.96, LGG GRADEI VS GRADEII: ACC=0.71
[70]	Outcome prediction of colorectal cancer	H&E	Extracted CNN features from epithelial tissue for prediction		
[24]	Thyroid cytopathology classification	H&E, RM and Pap	Fine-tuning pre-trained AlexNet		
[71]	Malaria, tuberculosis and parasites detection	Light microscopy	CNN-based patch classifier	Self-made [d]	Malaria: AUC=1; Tuberculosis: AUC=0.99; Hookworm: AUC=0.99
[72]	Gleason grading and breast cancer detection	H&E	The system incorporates shearlet features inside a CNN		
[73]	Metastases detection in lymph node	H&E	Ensemble of CNNs with hard negative mining	Camelyon 2016 dataset	Prediction of the automated system with the one of the pathologist: AUC=0.9948
[74]	Multiple cancer tissue classification	Various	Transfer learning using multi-Scale convolutional sparse coding	GBM and KIRC datasets	

[a] Breast Cancer histopathological samples from 162 women in the local hospital digitized via a whole-slide scanner at M:40x, resolution:0.25m/pixel

[b] High resolution histopathology images from 132 patients (size:10000x10000pixels)

[c] WSI on two types of brain cancer: glioblastoma multiforme (GBM) and lower grade glioma (LGG) from TCGA. Images were partitioned into tiles (size:1024 × 1024pixels, M:20×)

[d] (M:400×, GT by experts)

3.2.1 Challenges in Digital Pathology and Microscopy

Analyzing the literature it is evident that the organization of challenges focused on these topics has strongly improved the development of digital pathology analysis techniques. These contests provide not only the occasion to summarize the state of the art in some specific topics, but also provide very large and well-done datasets that usually remain open still after the end of the competition.
Some of these challenges are [26]: EM segmentation challenge 2012 for the 2D segmentation of neuronal processes, mitosis detection challenges in ICPR 2012 and AMIDA 2013, GLAS for gland segmentation in colorectal cancer tissue samples and, CAMELYON16 and TUPAC for the analysis of breast cancer tissue samples.
Dispite the difference in the proposed goals, in all these competitions CNNs were observed to be part of the better algorithms. The IDSIA team, for example, won both ICPR 2012 and the AMIDA13 challenges on mitosis detection with a CNN based approach [26, 27], but also the EM 2012 for 2D segmentation of neuronal processes. Also in this case they used the output of a CNN to compute pixel probabilities through mild smoothing and thresholding [26, 11].
Xu et al. team won the GLAS using three CNN in waterfall [26, 64]. The first one was used to classify pixels as gland or non-gland. Then from each feature map generated by the first CNN, a second one produces an edge map using the holistically nested edge technique. Finally, the last CNN merges informations about gland and edge maps to obtain the final segmentation.

CAMELYON16 was a very important event because it was the first time that a challenge provided a very large amount of data consisting of WSIs (whole-slide images) [26]. This allowed the possibility to train very deep model such as 22-layer `GoogLeNet` [75], 16-layer `VGG-Net` [76], and 101-layer `ResNet` [77]. On these models were based all the top-five performing systems of the CAMELYON16. The best solution, proposed by Wang et al. [73], achieved an AUC of 0.9935 which defeated the AUC of 0.966 of a pathologist who independently evaluated the test set. The architecture consists of an ensemble of two `GoogLeNet`, one trained with hard-negative and one without and uses the WSI standardization algorithm by Ehteshami Bejnordi et al. [78].

Also the team that won in all categories the last Tupac challenge [79] based his work on CNNs [26]. Their method locates those regions where the cell density is very high and uses CNNs to classify mitoses in the Region Of Interest (ROI). The result of this classification is then used as a feature vector associated to each WSI to fed a SVM that evaluate the tumor proliferation and molecular data scores.

3.2.2 Datasets in Digital Pathology and Microscopy

The most used datasets in digital pathology and microscopy are reported in the following paragraphs. Most of them have been developed for international challenges like the ones reported in the previous section, other are open-source database for research purposes.

MITOS Dataset This dataset was provided for the International Conference on Pattern Recogniction (ICPR) in 2014 [1]. The images were collected in Paris by the team of Professor Frédérique Capron, head of the Pathology Department at Pitié-Salpêtrière Hospital. He selected a set of breast cancer biopsy slides stained

[1] https://mitos-atypia-14.grand-challenge.org/dataset/

with standard hematoxylin and eosin (H&E) dyes and scanned by two slide scanners: Aperio Scanscope XT and Hamamatsu Nanozoomer 2.0-HT. In each slide, the pathologists selected several frames at X20 magnification located inside tumours and used them for scoring nuclear atypia. This was done dividing each X20 frame into four frames at X40 magnification which were used to annotate mitosis and to give a score to six criteria related to nuclear atypia. The number of frames is variable from slide to slide. In the training data set there are available 284 frames at X20 and 1136 frames at X40 magnification as RGB bitmap images in TIFF format.
Each 20X frame has a ground truth provided by two senior pathologists who worked independently and sometimes gave different score to the nuclear atypia. Score 1 corresponds to a low grade atypia, score 2 to a moderate grade atypia, and score 3 to a high grade atypia. For each 40X frame was annotated the list of mitosis (true and probably mitosis) and in case of disagreement a third opinion was asked for majority voting. Moreover three junior pathologists evaluated the nuclear atypia according to six criteria.

BCC Dataset The Basal Cell Carcinoma Dataset consists in 1417 small images of 300×300 pixels extracted from 308 RGB images of 1024×768 pixels [28]. Each image is obtained using a 10X magnification, stained with standard hematoxylin and eosin (H&E) and is related to an independent ROI on a slide biopsy. The ground truth was done by one pathologist who indicated the presence (or absence) of BCC and other architectural and morphological features (collagen, epidermis, sebaceous glands, ecrine glands, hair follicles and inflammatory infiltration).

ICPR 2014 HEp-2 Cell Datasets The ICPR 2014 HEp-2 Cell dataset consists in a training set composed by 13596 cell images that can be downloaded freely [39]. The test set is reserved to the contest organization to evaluate the challenging algorithms. The cell images were extracted from 83 images obtained using a monochrome high dynamic range cooled microscopy camera fitted on a microscope with a plane-Apochromat X20/0.8 objective lens and a LED illumination source. Each image is classified to one of six staining patterns: Homogeneous, Speckled, Nucleolar, Centromere, Nuclear Membrane, and Golgi.

ICPR 2012 Hep-2 Cell Dataset The ICPR 2014 HEp-2 Cell dataset consists of 1455 cell images extracted from 28 samples, which were acquired with a fluorescence microscope (40X magnification) coupled with 50-W mercury vapor lamp and a digital camera [39]. The dataset is divided into training set (721 images) and test set (734 images). The classes are six: Homogeneous, Coarse Speckled, Nucleolar, Centromere, Fine Speckled, and Cytoplasmic. Notice that two subcategories of ICPR2012 dataset (Fine Speckled and Coarse Speckled) are merged into one category (Speckled) in ICPR2014 dataset, and two less frequent patterns appearing in daily clinical cases, Golgi and Nuclear Membrane are introduced for developing more realistic HEp-2 cell classification systems.

ISBI 2012 dataset The ISBI 2012 dataset was provided for a EM segmentation challenge at ISBI 2012 and is still open for new contributions [31]. The training set is composed by 30 images (512×512 pixels) extracted from serial section transmission electron microscopy of the Drosophila larva ventral nerve cord (VNC). Each image owns a fully annotated ground truth segmentation map for cells (white) and membranes (black). The test set is publicly available, but its segmentation maps are

kept secret. The evaluation on the test set can be obtained by sending the predicted membrane probability map to the organizers.

MICCAI 2015 Warwick-QU Dataset The Warwick-QU dataset was build for the GlaS challenge (Gland Segmentation Challenge Contest) in MICCAI 2015 and consists of 165 images derived from 16 H&E stained histological sections of stage T3 or T4 colorectal adenocarcinoma [80], [81]. Each section belongs to a different patient, and sections were processed in the laboratory on different moments. Thus, the dataset shows an high inter-subject variability in both stain distribution and tissue architecture. The sections were digitized into WSIs using a Zeiss MIRAX MIDI Slide Scanner with a pixel resolution of 0.465 μm and then the WSIs were rescaled in order to have a pixel resolution of 0.620 μm which is equivalent to a X20 objective magnification. At the end, across the entire set of the WSIs, 52 visual fields were selected in order to cover the widest variety of tissue architectures possible. As ground truth a pathologist classified each visual field as benign or malignant, according to the overall glandular architecture, and segmented each glandular object (N.B. different glandular objects in a single image may be part of the same gland because a gland is a 3D structure which can appear fragmented in a 2D tissue section). The visual fields were further separated into smaller, non-overlapping images, whose ground truth is the same as the larger visual field. In the challenge, the dataset was split into training and test set an the ground truth was provided only for the training.

The Camelyon16 Dataset The Camelyon16 dataset consists of a total of 400 WSIs of sentinel lymph node from two independent datasets collected in Radboud University Medical Center (Nijmegen, the Netherlands), and the University Medical Center Utrecht (Utrecht, the Netherlands) [73] [2]. There are two different training set: the first one consists of 170 WSIs of lymph node (100 normal and 70 containing metastases) and the second 100 WSIs (60 normal and 40 containing metastases). The test set presents 130 WSIs. The ground truth consists of a pathologist's segmentation of metastatic lymph nodes regions and was provided only for the training set in two formats: XML files containing vertices, of the contours of the cancer metastases and WSI binary masks indicating the location of the cancer metastasis.

[2] https://camelyon16.grand-challenge.org/data/

3.3 Camelyon 2016

As seen in the previous section, one of the most important challenge in Digital Pathology was Camelyon 2016 which has been running for the past two years [3]. This challenge was particularly relevant because it was the first one using whole-slide images in histopathology.
This competition was organized by the 2016 IEEE International Symposium on Biomedical Imaging (ISBI-2016) and was highly successful with 32 submissions from as many as 23 teams

3.3.1 Motivation

In this challenge the focus was on the detection of micro- and macro-metastases in lymph node high-resolution digitized images [4]. Why analyzing lymph node? Because lymph node metastases occur in most cancer types (e.g. breast, prostate, colon) and acts as a kind of sentinel. In particular, lymph nodes are small glands that filter lymph, the fluid that circulates through the lymphatic system.

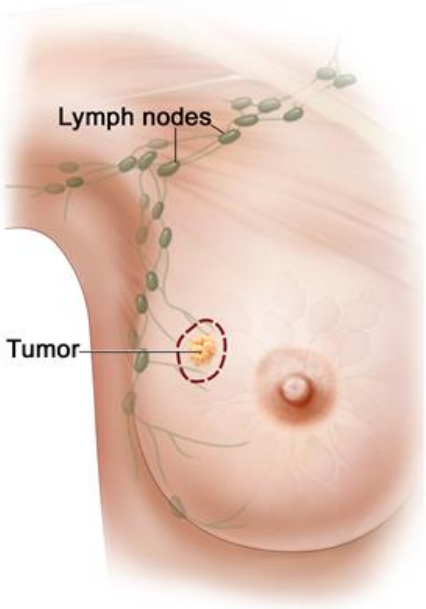


FIGURE 3.2: Metastatic axillary lymph node [25].

Axillary lymph nodes are the first place breast cancer is likely to spread like in figure 3.2. The metastatic involvement of the underarm area is known to progress regularly, from the first, via the second, to the third axillary level [82]. This process is one of the most important prognostic indicators in breast cancer, and of particular value in the choice of medical treatments. The more the cancer has spread to the lymph nodes, the poorer is the prognosis as can be seen in the graphic 3.3 [82].

The prognosis is gained from histological examination of all or most axillary nodes and the treatment of operable breast carcinoma almost always involves lymph-node dissection [82]. However, if a non-invasive or minimally invasive diagnostic procedure could provide accurate preoperative staging of the axilla, axillary dissection could be avoided in patients with no involved nodes [82].

[3] https://camelyon16.grand-challenge.org/home/
[4] https://camelyon16.grand-challenge.org/background/

Stage	Tumor Size	Lymph Node Involvement	Metastasis (Spread)
I	Less than 2 cm	No	No
II	Between 2-5 cm	No or in same side of breast	No
III	More than 5 cm	Yes, on same side of breast	No
IV	Not applicable	Not applicable	Yes

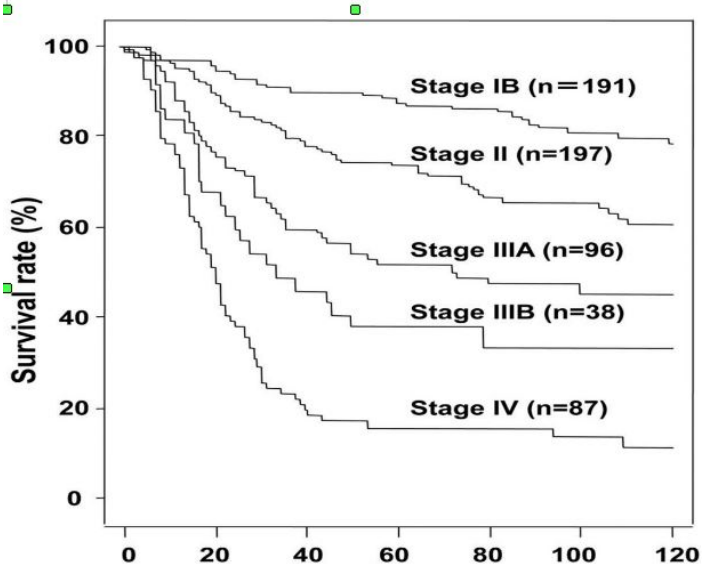


FIGURE 3.3: Prognostic indicators in breast cancer [25].

The diagnostic procedure for pathologists is, however, tedious and time-consuming and prone to misinterpretation [5]. An automated detection of lymph node metastasis would help pathologists by reducing their workload while at the same time would overcome the subjectivity and variability of diagnosis and obviously its cost.

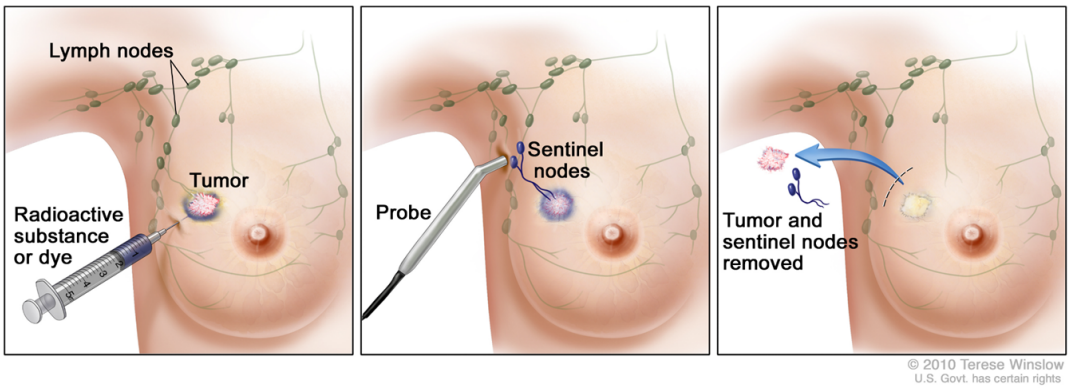


FIGURE 3.4: Breast treatment [25].

[5] https://camelyon16.grand-challenge.org/background/

3.3.2 Goal

The goal of the Camelyon Challenge was to evaluate new and existing algorithms for automated detection of metastases in hematoxylin and eosin (H&E) stained whole-slide images of lymph node sections of breast cancer patients [6].

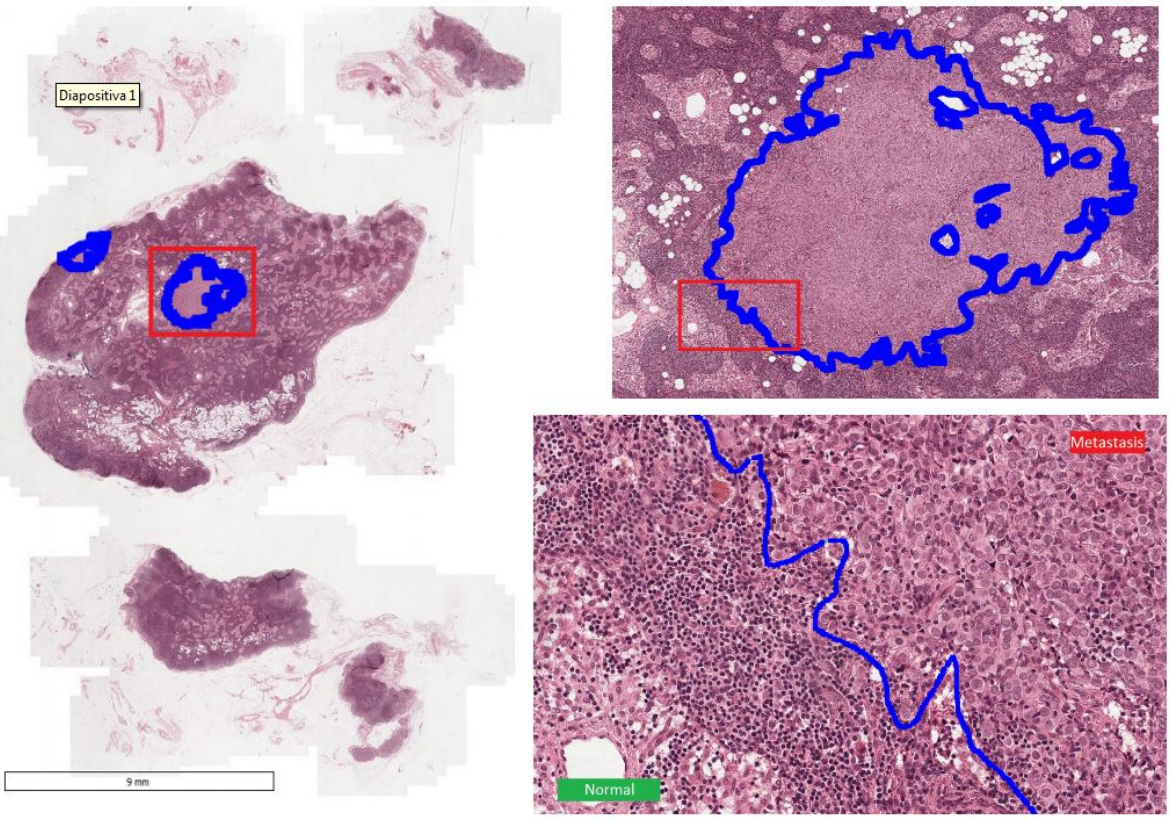


FIGURE 3.5: Low, Mid and High resolution images of a metastatic region [25].

3.3.3 Data

The Camelyon16 dataset consists of a total of 400 WSIs of sentinel lymph node from two independent datasets collected in Radboud University Medical Center (Nijmegen, the Netherlands), and the University Medical Center Utrecht (Utrecht, the Netherlands) [73] [7]. There are two different training set: the first one consists of 170 WSIs of lymph node (100 normal and 70 containing metastases) and the second 100 WSIs (60 normal and 40 containing metastases). The test set presents 130 WSIs collected from both Universities.

Institution	Train (Normal)	Train (Tumor)	Test
Radboud UMC	90	70	80
UMC Utrecht	70	40	50
Total	160	11	130

[6] https://camelyon16.grand-challenge.org/
[7] https://camelyon16.grand-challenge.org/data/

A pathologist segmented metastatic lymph nodes regions as ground truth and the average time for annotating each slide was 1 hour. This ground truth was provided only for the training set in two formats:

- XML files containing vertices, of the contours of the cancer metastases;
- WSI binary masks indicating the location of the cancer metastasis.

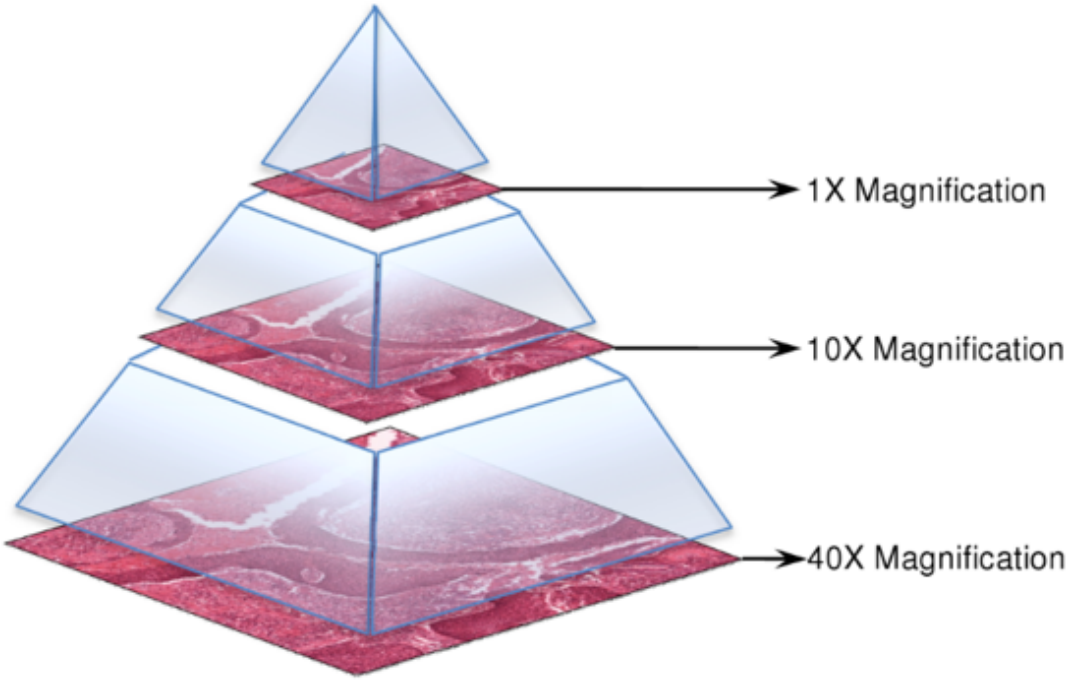


FIGURE 3.6: Multi resolution pyramid structure of the WSI format [25].

Whole-slide images are generally stored in a multi-resolution pyramid structure like the one in figure 3.6. Image files contain multiple downsampled versions of the original image. Each image in the pyramid is stored as a series of tiles, to facilitate rapid retrieval of subregions of the image.
Reading these images using standard image tools or libraries is a challenge because these tools are typically designed for images that can comfortably be uncompressed into RAM or a swap file. OpenSlide is a C library that provides a simple interface to read WSIs of different formats.

Automated Slide Analysis Platform (ASAP) is an open source platform for visualizing, annotating and automatically analyzing whole-slide histopathology images. ASAP is built on top of several well-developed open source packages like OpenSlide, Qt and OpenCV and is available on GitHub.

3.3.4 Evaluation

The two challenge datasets from Radboud UMC and UMC Utrecht were unified and divided into training and test sets, and the latter was made available when the ISBI 2016 challenge ended [8]. Evaluation of the algorithms for the manuscript submission had to be performed on the training data either through cross-validation or a

[8] https://camelyon16.grand-challenge.org/evaluation/

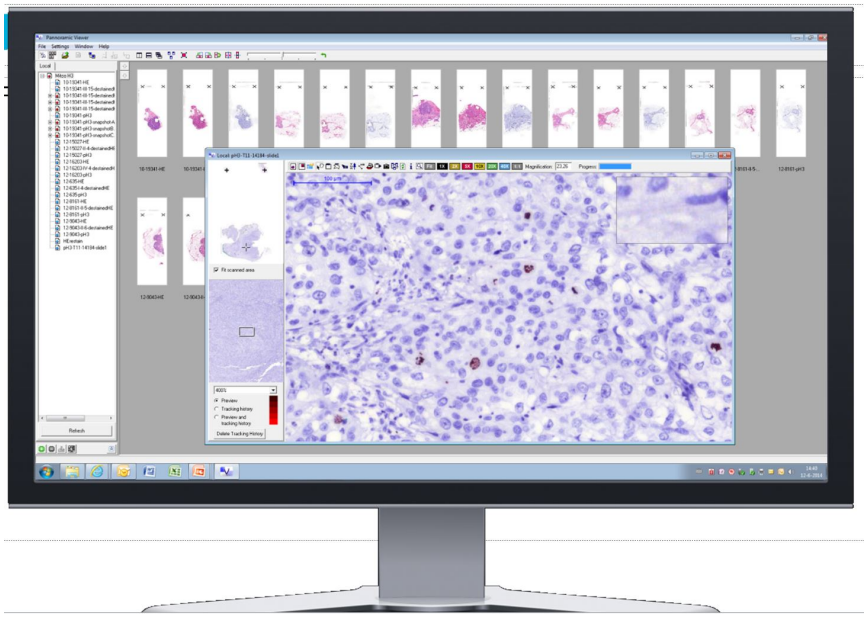


FIGURE 3.7: Example of tool for reading WSI [25].

hold-out experiment. Submissions to the competition were then evaluated on the following two metrics:

- Slide-based Evaluation: for this metric, teams were judged on performance at discriminating between slides containing metastasis and normal slides [73]. Each team submitted a probability for each test slide indicating its predicted likelihood of containing cancer. The competition organizers performed a receiver operating characteristic (ROC) analysis at the slide level and compared the algorithms using the area under the ROC curve (AUC) score.

- Lesion-based Evaluation: for this metric, participants submitted a probability and a corresponding $(x; y)$ location for each predicted cancer lesion within the WSI. The competition organizers performed a free-response receiver operating characteristic curve (FROC) which is defined as the plot of sensitivity versus the average number of false-positives per image. This is similar to ROC analysis, except that the false positive rate on the x-axis is replaced by the average number of false positives per image. In this challenge, the detected region was considered a true positive, if the location of the detected region was within the annotated ground truth lesion. If there are multiple findings for a single ground truth region, they were counted as a single true positive finding and none of them was be counted as false positive. All detections that were not within a specific distance from the ground truth annotations were counted as false positives. The final score gained by the second metric is defined as the average sensitivity at 6 predefined false positive rates: 1/4, 1/2, 1, 2, 4, and 8 FPs per WSI.

At the end two different final ranking were made, one based on the metastasis detection performance and one on the metastasis localization. The evaluation codes were provided both in Python and Matlab. The results are computed on the independent test set.

3.3.5 Results

The challenge was highly successful with 32 submissions from as many as 23 teams. Most of them were based on Deep Learning systems but there were also some frameworks based on classical image processing systems.

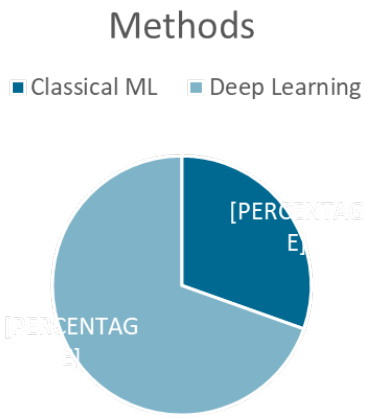


FIGURE 3.8: Statistic on the methods used for the Camelyon16 challenge [25].

Traditional Methods: Traditional image processing techniques used for this competition were:

- Texture and intensity based features
- Random Forest, SVM, Adaboost, etc.
- Analysis at multiple scales
- Cascade of Models operating at different magnifications

Deep Learning methods Thanks to the very large amount of WSIs provided for the competition [26], very deep model such as 22-layer `GoogLeNet` [75], 16-layer `VGG-Net` [76], and 101-layer `ResNet` [77] were trained. On these models were based all the top-five performing systems of the CAMELYON16.
Anyhow, it is important to notice that Deep Learning outperformed classical ML techniques. In fact, the first algorithm based on such methods is the one of the BioMediTech, University of Tampere in Finland which ranked on the 20th place (AUC=0.7612 for the first task and score = 0.2570 for the second task). The Deep Learning models proposed for this challenge are listed below:

- GoogLeNet (3)
- VGG Net (3)
- ResNet (1)
- AlexNet (1)
- U-Net (1)

- Multi-scale ConvNets (3)
- Fully Convolutional Network with CRF-RNN (1)
- Variant models Mostly not so deep Networks (3)

Classical pipeline

Almost all the algorithms based on Deep Learning models follow a general pipeline:

1. patches preprocessing
2. model training
3. patches classification
4. heatmap construction
5. metastasis localization

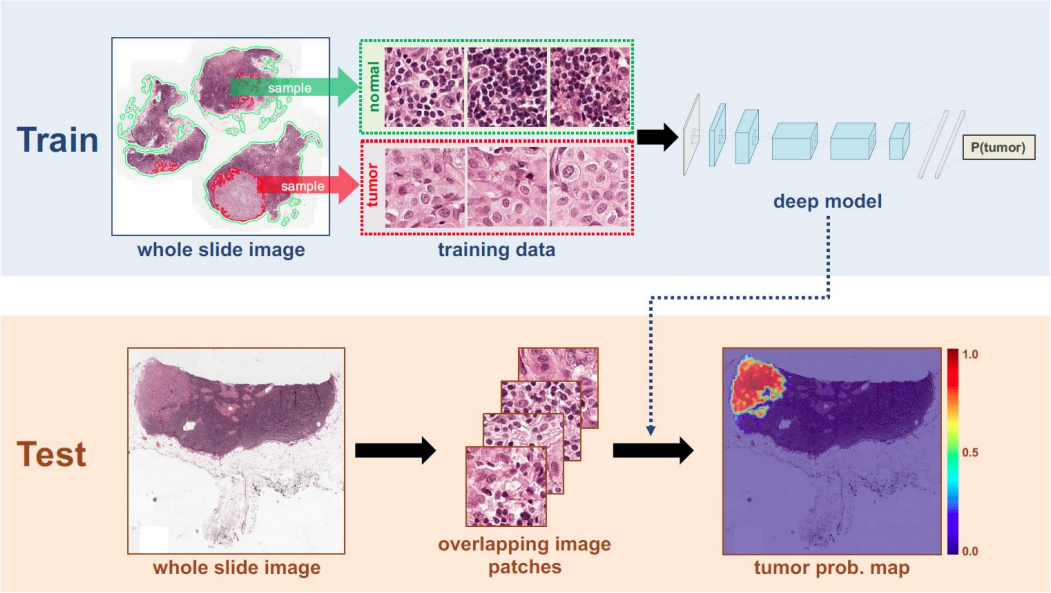


FIGURE 3.9: Pipeline of the best algorithm [73]

From the WSIs metastatic and non-metastatic patches are extracted and, sometimes, artificial patches are generated in order to reduce overfitting and enhance the training set.
Usually in the preprocessing phase the background is removed from patches in order to reduce the computational time (background is more than 80% of each patch) and noise. Many algorithms used the Otsu adaptive thresholding technique after transforming the RGB color space to the HSV color space.
Then the deep model is trained with the obtained patches and the corresponding ground truth. Sometimes pre-trained models are used for weights initialization and then a fine-tuning is made.
After the classification of each patch, all the prediction results are embedded into a unique heatmap image which highlights the tumor area. This can be done in several different ways (with or without overlap, etc). On these heatmaps, each pixel

contains a value between 0 and 1, indicating the probability that the pixel contains tumor.
The post-processing usually takes as input a heatmap for each WSI and produces as output a single probability of tumor for the entire WSI using an additional classifier. This classifier evaluates the morphological features of those areas whose probability of being tumor overcomes a certain threshold.

Public Leaderboard for WSI classification

The first task, i.e. the WSI classification, the AUC score was used. In figure 3.10 are listed the first twelve teams and the star indicates that the team has achieved an AUC value that surpasses the AUC of the pathologist. In graphic 3.11, the ROC curve of the first five teams is depicted.

Rank	Team	AUC	Submission date
01 *	**Harvard Medical School and MIT, Method 2 (updated)**	**0.9935**	**06 Nov 2016**
02 *	**Harvard Medical School, Gordon Center for Medical Imaging, MGH, Method 3**	**0.9763**	**24 Oct 2016**
03	Harvard Medical School, Gordon Center for Medical Imaging, MGH, Method 1	0.9650	07 Sep 2016
04	The Chinese University of Hong Kong (CU lab, Hong Kong), Method 3	0.9415	29 Aug 2016
05	Harvard Medical School and MIT, Method 1	0.9234	01 Apr 2016
06	EXB Research and Development co., Germany	0.9156	01 Apr 2016
07	The Chinese University of Hong Kong (CU lab), Hong Kong, Method 1	0.9086	08 June 2016
08	Harvard Medical School, Gordon Center for Medical Imaging, MGH, Method 2	0.9082	24 Oct 2016
09	The Chinese University of Hong Kong (CU lab), Hong Kong, Method 2	0.9056	20 July 2016
10	DeepCare Inc, China	0.8833	05 Nov 2016
11	Independent participant, Germany	0.8654	01 Apr 2016
12	Middle East Technical University, Departments of EEE, NSNT and HS, Turkey	0.8642	01 Apr 2016

FIGURE 3.10: Public Leaderboard for WSI classification; first 12 positions [25].

First algorithm The best solution, proposed by Wang et al. [73], achieved an AUC of 0.9935 which defeated the AUC of 0.966 of a pathologist who independently evaluated the test set. The architecture consists of an ensemble of two `GoogLeNet`, one trained with hard-negative color-normalized patches extracted from each WSI and one without hard-negative. The average of the two prediction results was taken as the final prediction value. The algorithms uses the WSI standardization algorithm by Ehteshami Bejnordi et al. [78] which standardize all the images to have the same staining color. This first phase generates a sort of heatmap which represents the probability of metastatic lesions. Then a random forest classifier is trained with the features extracted from the probability map in order to localize the lesions.

Second algorithm The second team trained a fully convolutional ResNet-101 network with atrous convolution and atrous spatial pyramid pooling which enlarge the field-of-view for prediction and allow capturing objects as well as image context

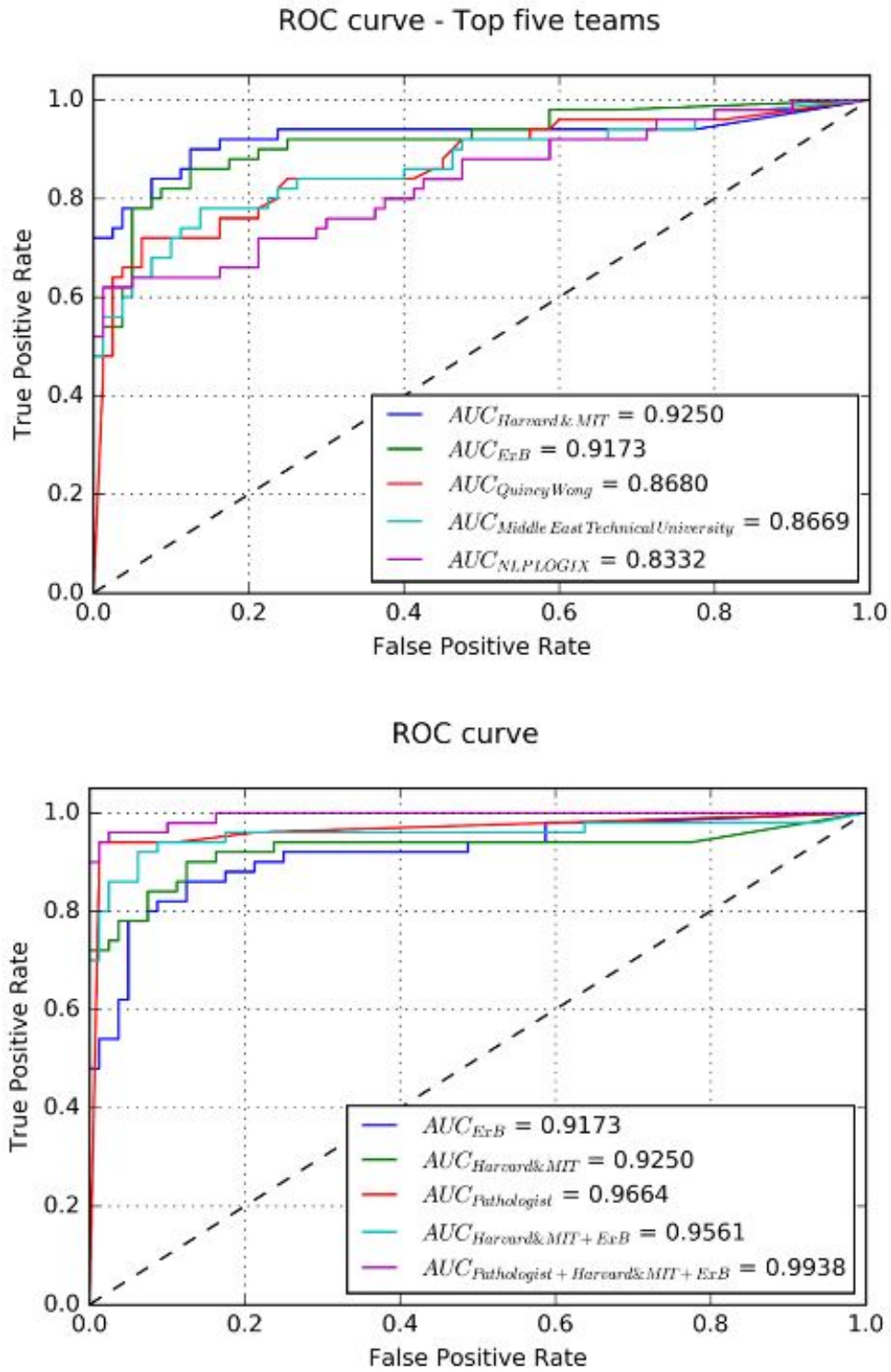


FIGURE 3.11: ROC of WSI classification; first 5 positions and ROC of the first 5 WSI classification related to the AUC reached by a pathologist [25].

at multiple scales [9]. Patches of size 512*512 at level 1 were used as input to the network. For each mini-batch 10 normal and 10 tumor patches were randomly taken to train the network. The classification task is based on features extracted from tumor probability map with a random forest classifier as described in the submission by HMS-MIT Method 1. Further postprocessing step was used to remove candidates that are too small for the lesion-detection task.

Third algorithm This team reproduced the algorithm of the top-performing team but, instead of training the GoogLeNet from scratch, they used the pre-trained GoogLeNet model trained for ImageNet classification to initialize the weights and then fine tuned the network. Then similar strategies were taken to analyze the lesion levels.

Fourth algorithm This team used the pre-trained VGG-16 net for initializing their final fully convolutional network [10]. To avoid over-fitting, they extracted the patches (on-the-fly to save memory) randomly from the WSI to augment the training dataset.The post-processing step includes outliers suppression through a median filter and metastasis localization in the centroids of connected components on the mask after thresholding. The WSI classification results were simply generated based on the location results by selecting the maximum value of whole metastasis probability map

Fifth algorithm This team trained a 22-layer GoogLeNet using tumor and normal patches [11]. Then they applied the trained model to partially overlapping patches from each WSI to create tumor prediction heatmaps. After evaluating the accuracy of the heatmaps, they extracted additional training patches from the false positive regions in order to train a final model. For the slide-based tumor classification task, they extracted a set of geometric features from each heatmap in the training set, and trained a random forest classifier to estimate the probability that each slide contained metastatic cancer. Finally they applied these models (CNN and random forest classifier) to the test images to provide a slide-based estimate of the probability of cancer metastases. For the lesion-based tumor region segmentation task, a threshold of 0.90 to the tumor probability heatmaps was applied and the tumor location was assigned to the center of each predicted tumor region.

[9] https://camelyon16.grand-challenge.org/PerTeamResult/?id=MGH3
[10] https://camelyon16.grand-challenge.org/PerTeamResult/?id=CULAB3
[11] https://camelyon16.grand-challenge.org/PerTeamResult/?id=BIDMC_CSAIL

Warwick-QU Team Approach

The Warwick-QU Team from the Warwick University, Warwickshire (UK), presented a project based on a variant of the U-Net convolutional-deconvolutional neural network that was explained in subsection 2.3.3 [12].
First of all they removed the background from all images by segmenting the tissue regions (ROI, i.e. region of interest) from fatty and white background areas in order to optimize the sequent operations like in figure 3.12. This operation was done using a simple fully convolutional network (FCN) with a single up-sampling layer. Then they performed a stain normalization in order to deal with stain variation in the H&E stained slide images.

FIGURE 3.12: Removal of the background from all images by segmenting the tissue regions (ROI, i.e. region of interest) from fatty and white background areas (image extracted from the Warwick-QU slides in the results section of [25]).

UNet architecture was implemented using `Tensorflow` in order to allow an easy customization of the network for the cancer segmentation purpose. The output of such network consists of a probability map, whose pixel intensity values express the probability of being part of a cancer metastasis as can be seen in figure 3.13. The probability of belonging to a cancer area of each pixel is further elaborated by considering the probability values within a region and the region area. Finally, the candidate regions are further refined using the weighted probability map.
The training dataset was composed by 20000 RGB patches of size 428×428 at magnification level 2 (10X). From those 20000 patches, 120000 were taken from normal

[12] https://camelyon16.grand-challenge.org/PerTeamResult/?id=Warwick

WSIs and 8000 from metastatic ones. The whole dataset was divided in order to perform the training and the validation (90% of the whole dataset for the training set and 10% for the validation set). The network was trained for more than 50 epochs with batch size 10 with cross-entropy as the cost function.
The team reached the 16th position in the WSI classification leaderboard and the 18th position in the tumor localization leaderboard. The results of the method are shown in figure 3.14.

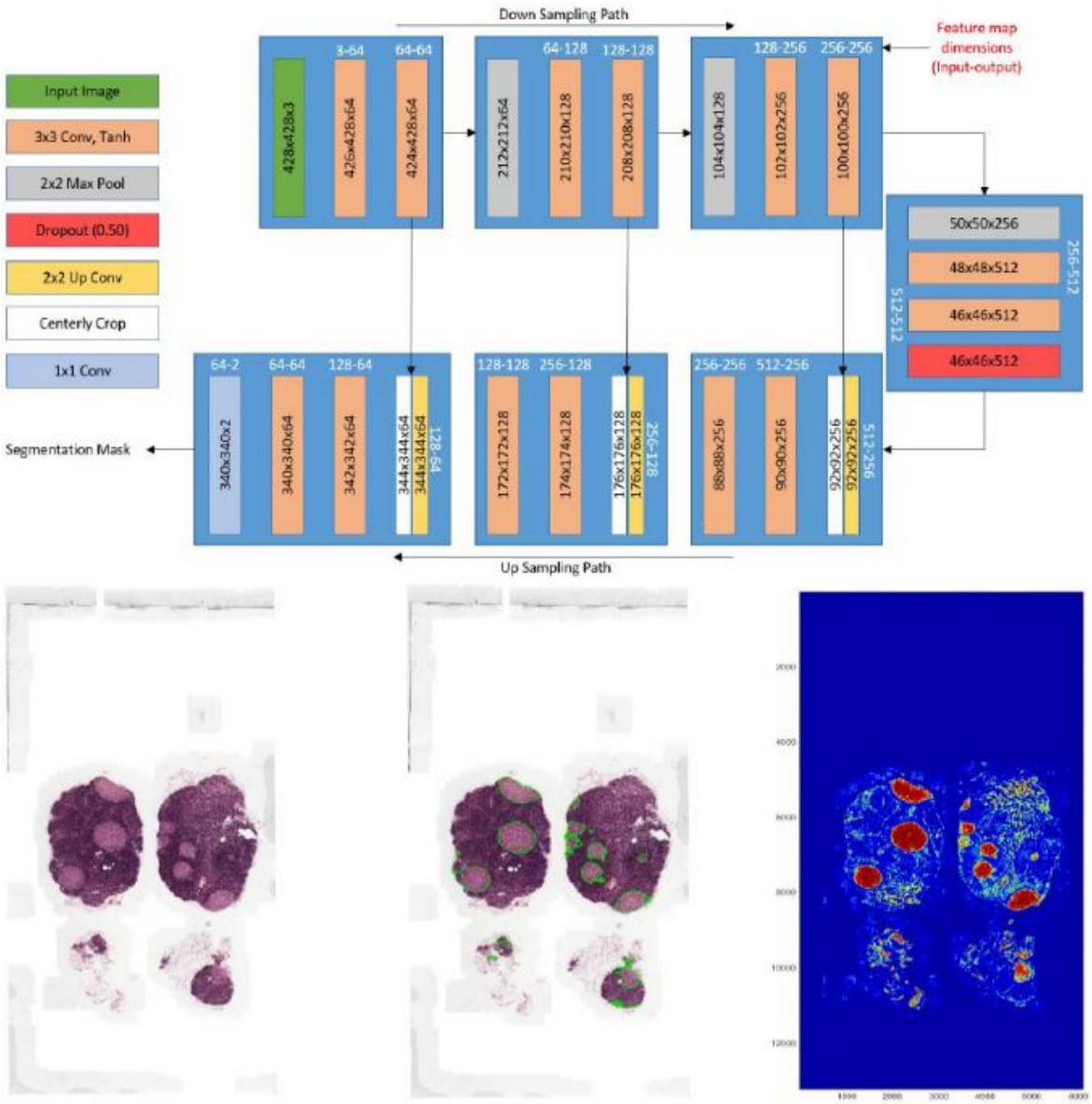
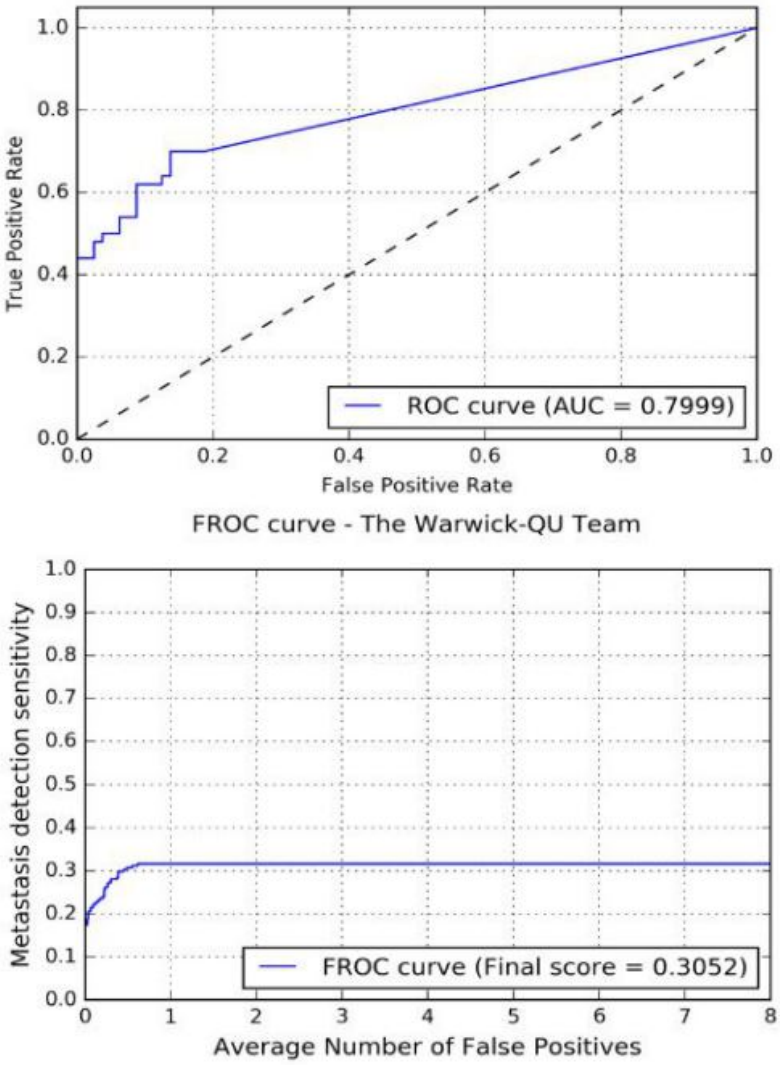


FIGURE 3.13: UNet customization by the Warwick-QU team (image extracted from the Warwick-QU slides in the results section of [25]).



FPs/WSI	0.25	0.5	1	2	4	8
Sensitivity	0.262	0.307	0.316	0.316	0.316	0.316

FIGURE 3.14: The three figures show respectively the ROC curve, the FROC curve and the average sensitivity of the developed system at 6 predefined false positive rates: 1/4, 1/2, 1, 2, 4, and 8 FPs per WSI (image extracted from the Warwick-QU slides in the results section of [25]).

3.4 MICCAI 2015 - Gland Segmentation Challenge Contest

A second important challenge in Digital Pathology was GlaS, i.e. Gland Segmentation Challenge which was held in conjuction with MICCAI 2015, Munich, Germany [13]. The aim of this contest was to focus on the gland segmentation problem and to validate state-of-the-art algorithms on the same standard dataset.

3.4.1 Motivation

In this challenge the main objective was to segment glands [14], i.e. fundamental histological structures which are able to synthesize and secrete proteins and carbohydrates (e.g. hormones) into the bloodstream (endocrine glands), into internal cavities of the body or directly out of the external surface of the body (exocrine gland) [15].

Usually glands present a tubular structure in which it can be distinguished the internal lumen surrounded by epithelial cells as shown in Figure 3.15 The particular arrangement of epithelial nuclei around the lumen can be used to identify clearly glands structures [80].

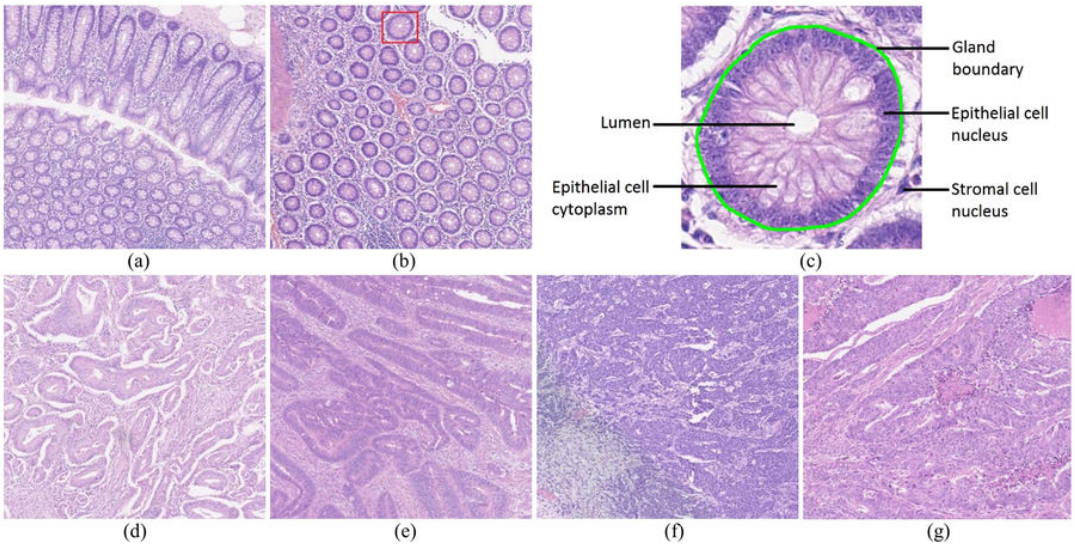


FIGURE 3.15: Hystopathological images of glands affected by different grades of tumor. (a), (b) normal tissue, (c) particular of the structure of a normal gland taken from the red rectangle in (b), (d) and (e) low grade tumor, (f) and (g) high grade tumor [83].

These cells structures are present in most of the organ systems and are really important for diagnosis of several adenocarcinomas [80]. In fact, previous studies show that the morphology of glands can be used for grading of prostate, breast and colon adenocarcinomas [80] which are really widespread form of cancer.

In the dataset collected for the GlaS challenge, colon images are involved. Also in this case, the morphological characteristics of the glands structure reflects the stage and the severity of the cancerous lesion [80]. In Figure 3.15 can be seen examples of normal tissues, low grade adenocarcinomas which still exhibits glandular structures similar to a healthy gland and high grade adenocarcinomas where glandular structures are definitely degenerated. Therefore, an automatic segmentation of glands

[13] https://warwick.ac.uk/fac/sci/dcs/research/tia/glascontest/
[14] https://warwick.ac.uk/fac/sci/dcs/research/tia/glascontest/about/
[15] https://en.wikipedia.org/wiki/Gland

and the subsequent statistics of their morphology can help pathologists in grading tumors.

3.4.2 Goal

In goal of this challenge, is to collect state-of-the-art gland segmentation algorithms on images of Hematoxylin and Eosin (H&E) stained slides consisting of a variety of histological grades [16]. The novelty of this challenge is that it focuses on various grade of cancers (healthy, benign, intermediate and high)

The main advantage of such a challenge is the fact that all teams could train their algorithms and validate them on a standard dataset provided by the organization, allowing a clear comparison and ranking between them.

3.4.3 Data

The GlaS dataset consists of 165 images derived from 16 H&E stained histological sections of stage T3 or T4 colorectal adenocarcinoma [80], [81]. Each section belongs to a different patient, and sections were processed in the laboratory on different moments. dataset. Thus, the dataset shows an high inter-subject variability in both stain distribution and tissue architecture. The sections were digitized into WSIs using a Zeiss MIRAX MIDI Slide Scanner with a pixel resolution of 0.465 μm and then the WSIs were rescaled in order to have a pixel resolution of 0.620 μm which is equivalent to a X20 objective magnification. At the end, across the entire set of the WSIs, 52 visual fields were selected in order to cover the widest variety of tissue architectures possible. As ground truth a pathologist classified each visual field as benign or malignant, according to the overall glandular architecture, and segmented each glandular object (N.B. different glandular objects in a single image may be part of the same gland because a gland is a 3D structure which can appear fragmented in a 2D tissue section). The visual fields were further separated into smaller, non-overlapping images, whose ground truth is the same as the larger visual field. In the challenge, the dataset was split into training and test set an the ground truth was provided only for the training.

3.4.4 Evaluation

The evaluation of each proposed algorithms was based on the following criteria [81]:

- accuracy of the detection of individual glands (F1 score computed on the detected objects);
- volume-based accuracy of the segmentation of individual glands (Similarity measures like Object-Level Dice Index and Adjusted Rand Index);
- boundary-based similarity between glands and their corresponding segmentation (object-level Hausdor distance)

Sometimes the volume-based segmentation accuracy involves the boundary-based segmentation accuracy between a gland and its segmentation but it is not true all the times [81]. In fact, the first metric is defined using the label assigned by the segmentation algorithm to each pixel while the second is defined using the boundary of each gland traced by the same algorithm. Thus, while the pixels labels may be correctly assigned, the boundary curves may be very different.

[16] https://warwick.ac.uk/fac/sci/dcs/research/tia/glascontest/about/

3.4.5 Results

In this section are described the first seven positions of the final ranking. All the informations reported below are taken from [81].

CUMedVision The method of the CUMedVision is based on a FCN which outputs directly a segmentation probability map and an image containing the contours of glands separately. This was done using two different branches in the up-sampling part of the network, one for the probability maps and one for the contours maps while the down-sampling path is shared. This multi-task learning approach allowed the separation of touching glands. To overcome the lack of enough data for training, the down-sampling part of the network was initialized with the weights of a public available model from DeepLab trained on the 2012 PASCAL VOC dataset5.
The CUMedVision2 submitted model ranked 1^{th} and considers both gland objects and contour masks. The first version of the team, CUMedVision1, was based only on gland objects mask and ranked 5^{th}.

ExB The ExB team used a multi-path CNN where each path presents a different configuration of the convolutional layers in order to capture different features and give a different point of view over the problem. All the paths were connected to the same set of two FC layers interconnected by a leaky rectified linear unit as activation function followed by a softmax layer. The training was performed using a stochastic gradient descent with momentum, randomly initialization and a step-wise learning rate schedule, data augmentation and zero mean and unit variance normalization over the training data. The experimental analysis has shown that more than three paths caused overfitting (the dataset is too small).
The simple-path networks task was the detection of borders of glands (ground truth built optimizing the F1 score and the object-Dice index) and its output was used together with the output of a binary classifier (2 CONV and 1 FC layers) which discriminate between benign and malignant classes starting from the row image. The post-processing parameters were different according to the predicted class. In the postprocessing they cleaned noise, filled holes and removed spurious structures through thresholding.
The ExB team submitted three proposals: ExB1 (ranked 2^{th}) is the approach described up to now, ExB2 (ranked 6^{th}) is the same approach without the border detection network, ExB3 (ranked 3^{th}) is the same approach without any post-processing.

Image Analysis Lab Uni Freiburg The Uni Freiburg team used UNet convolutional-deconvolutional neural network described in Section 2.21 to perform the segmentation. The network takes as input a raw RGB image and returns a binary mask which separates the glands from the background in an end-to-end fashion.
UNet was trained from scratch using an augmented dataset obtained through random elastic deformations, rotations, shifts, flips, color transformations and blurs. The parameters for training were taken from the original paper of UNet [10] with the only exception of more training iterations and a slower decrease of the learning rate. As [10], to separate touching glands, they added a very high pixel-wise loss weight for pixels which are between two really close objects.
The first submission of the team ranked 7^{th} and applies to the UNet output only a connected component labeling, while the second, which ranked 4^{th}, applies to the output a morphological hole-filling and deletion of objects smaller than 1000 pixels.

3.5 ISBI 2012 - 2D EM segmentation challenge

This challenge is the first one in the field of bi-dimensional segmentation of neuronal structures in electron microscopic (EM) brain images and was held during the ISBI 2012 international conference in Barcelona, Spain. This event was organized to stimulate the progress in the field of automatic reconstruction of neural circuits through a crowd-sourcing approach. All the informations of this section, except where the source is specified, are taken from [84] and [17].

3.5.1 Motivation

The structure of the brain is still mainly unknown. For overcoming this lack of knowledge many engineering techniques have been developed in recent years. One branch of research of neuroanatomy is trying to map the synaptic connections of the brain system [11]. Studies have been conducted both on invertebrate and mammalian brain samples [84]. This is a really ambitious and challenging task which requires the collaborations of clinicians and engineers. The main tools used for this purpose is serial-section Transmitted Electron Microscopy (ssTEM) which is able to resolve individual neurons and their shape starting from neural tissue sections of the magnitude order of *nm* [11].
The segmentation of neural structures from these high-resolution images is really time consuming and unfeasible for a human. Some semi-automated methods have been developed to reduce the human efforts but for large volume such a way is still unfeasible [84]. Therefore it is necessary to find a completely automated solution for segmenting neural structures and connections of 3D neural tissue sample avoiding gradually any human interaction. In that sense, this challenge tried to accelerate research in this field through a crowdsourcing approach [84], which enables to compare different algorithms on the same dataset with the same metrics.

3.5.2 Goal

The goal of this competition was to perform a particular type of semantic segmentation, which can be referred as *boundary segmentation*. The applicative aim is to segment the boundaries between neurites (neuron membranes) starting from grayscale EM images by assigning label 1 to the pixels located inside cells and 0 to the pixels of the boundaries. This task is particularly challenging because many boundaries are not so well defined and can be misunderstood with the ones belonging to intracellular structures like mitochondria and vesicles.

3.5.3 Data

The ISBI12 dataset consists in a set of ssTEM images of the Drosophila larva ventral nerve cord extracted from the dataset of [85].
In Figure 3.16 [85] (A) is represented a cross-section of one Drosophila larva brain hemisphere: primary neurons with their axon fascicles are depicted in purple. A detail of the neuropile, which is formed by neurites, i.e. neuronal processes, is proposed in Figure 3.16 (C) (scale bar $1\mu m$). The arrow inside (C) underlines a synapse which is zoomed in Figure 3.16 (D) (scale bar $350nm$) while in (D) the right arrow underlines a presynaptic vesicles and the left one an obliquely sectioned microtubule.

[17] http://brainiac2.mit.edu/isbi_challenge/

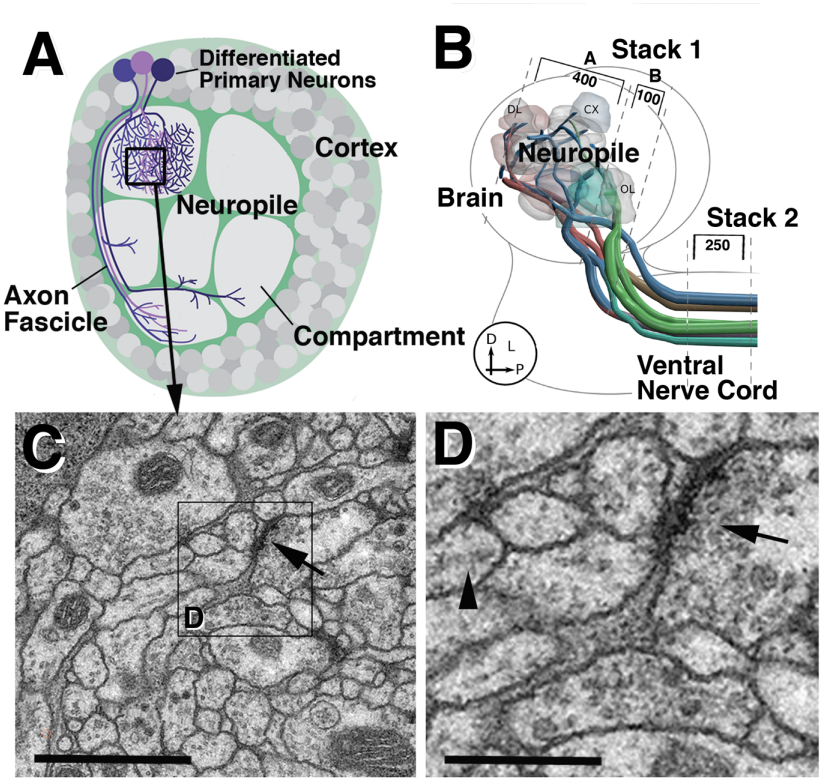


FIGURE 3.16: (A) cross-section of one Drosophila larva brain hemisphere, (B) its brain and ventral nerve cord, (C) detail of the neuropile (scale bar $1\mu m$), (D) details of a synapse (scale bar $350nm$) [85].

Instead, in Figure 3.16 (B) is represented a larval brain and ventral nerve cord. Referenced as Stack 1 and Stack 2 can be find a schematic of the positions and orientations used for acquiring the serial section. The first stack contains two sets of neuropile sections (one of 400 and the other of 100 sections), while the second stack contains 250 sections of the ventral nerve cord. The axons that connect brain and ventral nerve cord are depicted using colored lines.

In our case, the sections were acquired through a FEI electron microscope driven using Leginon and equipped with a Tietz camera and a goniometer-powered mobile grid stage [84]. The magnification is set to $5600\times$ binned at 2, leading to a $4 \times 4nm$ per pixel resolution.

This type of image acquisition generates a volume of images which is highly anisotropic in terms of axis resolution, i.e. the resolutions along the x and the y axes are really high while the one along the z axis is really low due to the physical difficulties of sectioning the tissue block. Thus the obtained images are 2D projections of the whole section and the membranes are likely to be orthogonal to the cutting plane to look good.

The training set, for example, is stored in a unique .tiff volume consisting in 30 consecutive images of 512×512 pixels. The total volume size is $2 \times 2 \times 1.5\ \mu m$, with a resolution of $4 \times 4 \times 50nm/pixel$. These images contain lots of noise and alignment errors as they have to represent images acquired in a typical real-world setting. Also the test set contains 30 grayscale images.

The ground truth is publicly available only for the training set. The test set one, instead, was kept secret and used by the organizers to evaluate the algorithms and the predicted test masks submitted by the teams.

The ground truth of the training set was created by segmenting each 2D plane by one coauthor (through the open-source software TrakEM2) while the ground truth of the test set was created by other two coauthors who worked independently from each other (one through the open-source software VAST3, the other through TrakEM2). The definitive test labels were obtained by comparing the two labels obtained by the coauthors and correcting manually the disagreements and guaranteeing the coherence and continuity of segmentation in the .tiff volume (avoiding disagree in objects split or merge usually).

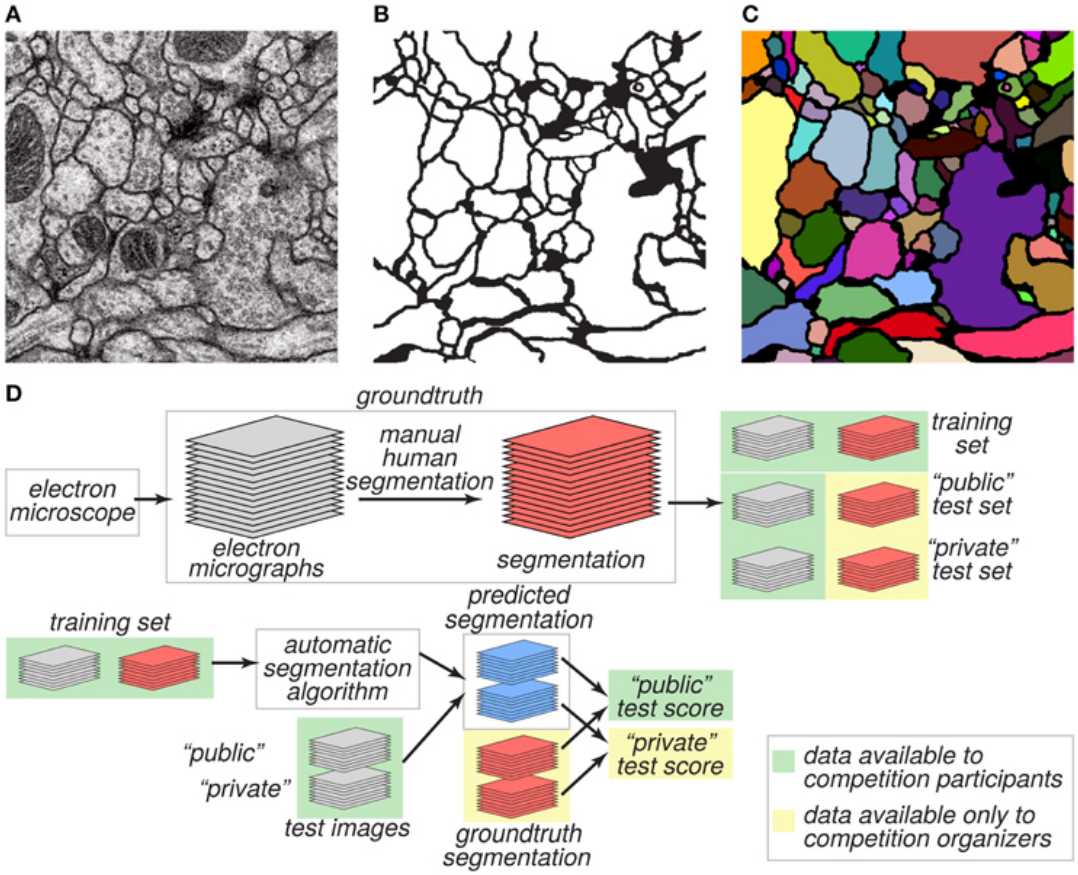


FIGURE 3.17: Graphical description of the ISBI12 dataset [84].

3.5.4 Evaluation

Participants had to submit the prediction masks obtained by the publicly available test set which are then compared through some metrics with a ground truth provided by an human expert as explained in section 3.5.3. Each submission was first evaluated and ranked using bi-dimensional topology-based segmentations metrics and the pixel error [18]. The two topology metrics were the minimum splits and merge warping error (metric that penalized topological disagreements like object splits and merges) and the foreground-restricted rand error (metric that measures the similarity between two segmentations computes as 1-max(F score) where F score is referred to the F-score of the foreground-restricted Rand index excluding the background). Despite a first ranking was done using these metrics, this scoring system has shown

[18] http://brainiac2.mit.edu/isbi_challenge/evaluation

to be not robust to variable widths of neurite borders. After that, the organizer some new evaluation metrics that best fitted their segmentation task:

- V^{Rand}*(thinned)*: Rand Scoring after border thinning;
- V^{Info} *(thinned)*: Information Theoretic Scoring after border thinning.

These metrics are particular normalized versions of the Rand error and Variation of Information metrics. These two metrics are restricted only to the foreground, i.e. they do not consider the zero labels which corresponds to the background pixels of the ground truth. For a deeper insight in these metrics see [84].

3.5.5 Results

The complete ranking lists of the challenge can be found at [84]. The organizers say also that the results have probably saturated due to the restricted dataset and the inherent ambiguities of the evaluation metrics [84].
At the deadline of the competition, only seven team made their algorithms public at the conference and their brief description can be found in the additional material provided by [84].

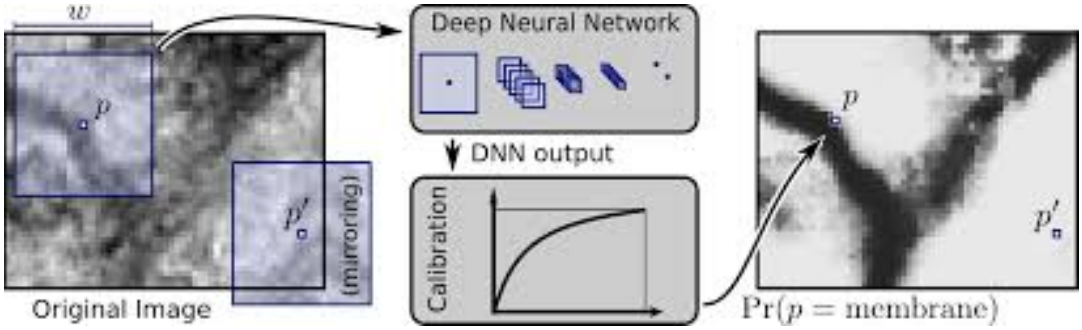


FIGURE 3.18: Pipeline of the winning approach [11].

The winner was the IDSIA team [11] with the deep learning approach shown in Figure 3.18. It was the first time that this team faced EM images. They proposed a CNN model that acts as a single pixel classifier into the membrane or non-membrane classes [11]. The model computes the probability of the pixel to belong to a certain class starting from a row crop of the original image centered in that pixel. The final probability map of a given image is obtained by classifying all the pixels of the image. The final segmentation mask is then obtained by a weak post-processing consisting in a monotone cubic polynomial calibration function (like a grayscale transformation) to avoid an overrate of the membrane probability, a smoothing through a median filter (2 pixel radius) and thresholding.
This approach wins in all three metrics and, considering the pixel error, is the only method that outperforms also humans [11]. Moreover, all the other methods, even if using machine learning approaches, were based on hand-crafted features [84].

Chapter 4

Methods and Experimental Set-up

4.1 General Pipeline

Within this thesis, the possibility of applying Transfer Learning and Data Augmentation in the field of Semantic Segmentation of histopathological images is evaluated. This is done by choosing a deep convolutional model (UNet), training it on a really big histopathological dataset (Camelyon16) and transferring the acquired knowledge to two new tasks with increasing difference from the original one (GlaS and then ISBI12). Expressly no postprocessing is performed in order to better compare the improvements/deterioration due to Transfer Learning and Data Augmentation. The general pipeline followed in this experiment is:

1. Training UNet with Camelyon16 Data
2. Fine-tuning of UNet with augmented GlaS Data
3. Fine-tuning of UNet with augmented ISBI12 Data

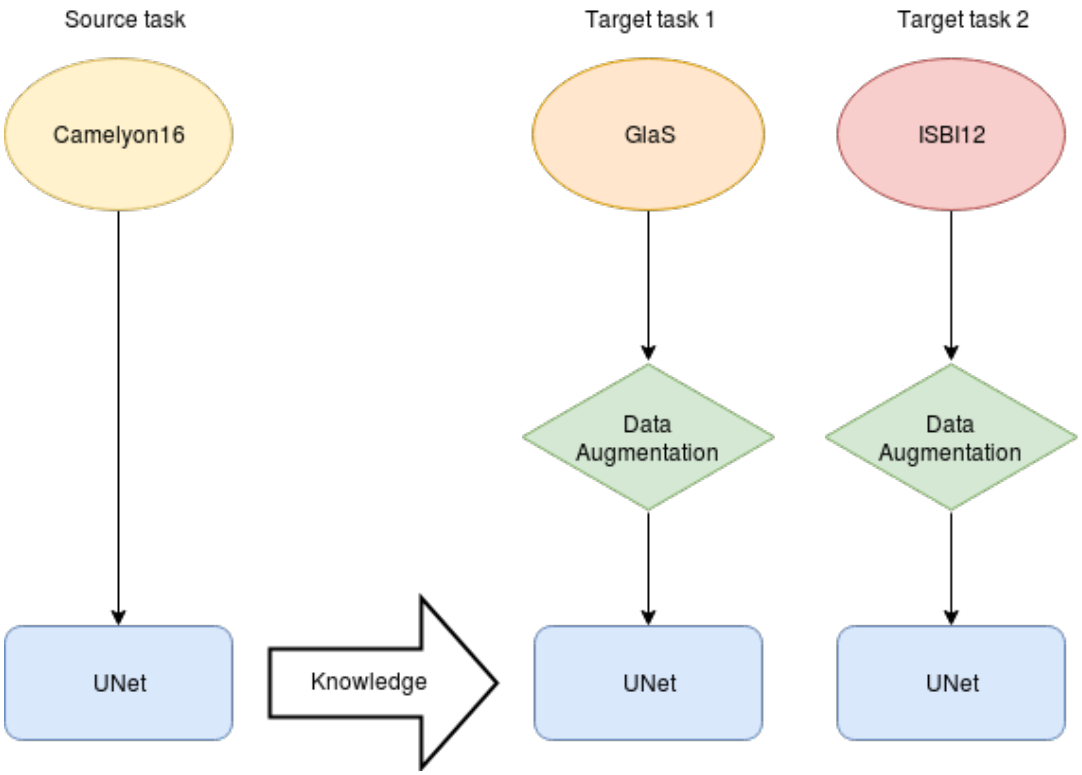


FIGURE 4.1: General pipeline followed in this thesis

4.2 Methods

4.2.1 OpenSlide

`OpenSlide-Python` is a Python library [1] which acts as a wrapper for the original `C OpenSlide` library [2]. This `C` library allows the users to easily read Whole Slide Images (WSIs), i.e. high-resolution images that require tens of gigabytes when uncompressed that can not be read using standard image tools [86] [87].
Moreover, a single WSI usually contains different resolution of the same images to provide a fast navigation and zoom. In the case of Camelyon16 dataset, for example, each WSI contains multiple downsampled versions of the original image. Then, each image in the pyramid is stored as a series of tiles, to facilitate rapid retrieval of specific subregions. In this case, `Openslide` allows reading a specific subregion at a resolution which is closest as possible to the desired zoom level.
Therefore, this library is frequently used in the histopathological field for building tools and GUIs for clinicians.
The library can read the following image formats [86]:

- Aperio (`.svs, .tif`);
- Hamamatsu (`.ndpi, .vms, .vmu`);
- Leica (`.scn`);
- MIRAX (`.mrxs`);
- Philips (`.tiff`);
- Sakura (`.svslide`);
- Trestle (`.tif`);
- Ventana (`.bif, .tif`);
- Generic tiled `TIFF` (`.tif`).

There are lots of possible image formats and no standard for WSIs because each microscopy company wants the customers to use their formats, libraries and viewers (typically only for Windows) to read images acquired using their instruments [87]. In my opinion, this is the main reason why companies usually do not document their formats and, if they do it, the documentations lacks of many fundamentals details. Despite all, some common characteristics of these different formats are:

- based on `TIFF`;
- quick access to lower resolutions;
- optimized for random access ;
- use of different lossy compression paradigms;
- effectively unbounded in width and height;
- presence of metadata.

[1] https://openslide.org/api/Python/
[2] https://openslide.org/

The library can deal with different WSI formats and proposes to the end-user an abstract interface that is independent of the format or the vendor [87]. This is done hiding and overcoming all the format-specific metadata and operations. For example, `Openslide` extracts only uncompressed RGBA data avoiding operations for the decompression of data which are strictly connected to the format. Moreover, the library avoids any image scaling operation. In particular, it provides the following features [87]:

- a read-only interface for extracting rectangular ROIs;
- image representation as an ordered list of layers where:
  - layer 0 corresponds to the maximal resolution
  - each layer is the downsampled version of the previous one
- function that generates individual Deep Zoom tiles from slide objects that allows[3]:
  - Static tiling: create all the tiles
  - Dynamic tiling: creates the tiles on demand (useful for web applications)
- function that computes the downsampling factor for each layer with respect to the zero one;
- function that computes the next largest layer for a given downsamping factor;

4.2.2 Augmentor

`Augmentor` is a Python package whose primary purpose is Data Augmentation and Processing for machine learning (all information are taken from [88]). The latest version of `Augmentor` uses multi-threading to speed up the generation of images when dealing with large datasets.
This library consists in a set of classes that perform image processing operations (rotation, crop, resize and so on) that can be chains together sequentially in order to create a multi-stage pipeline. When images are processes through this pipeline, these operations are applied stochastically according to a probability parameter decided by the user for each operation. Also, the operations parameters (e.g. rotation range) are randomly set within a range specified by the user. This pipeline generates a set of augmented images whose size depends on the number of operations in the pipeline, and the range of values available for each operation. At the end, the user can sample from the generated images how many images he wants.
The transformation that can be applied in such case can range from space transformation like elastic distortions, perspective transform, translation, rotation, warping, vertical or horizontal mirroring, scaling, color space shifts, changing the brightness, pads, crops, etc. These transformations can be easily traced also in the segmentation masks, while in the case of classification tasks this would not be a problem.
This library allows doing data augmentation on the fly, i.e. during the training without saving the new images through a generator. These generators return augmented data and corresponding labels indefinitely and can be given directly as input to the fit function. In this case the training phase is a little bit longer but it solves lots of memory problems.

[3] https://github.com/openslide/openslide/issues/149

4.2.3 Keras

Keras is a high-level neural networks Application Program Interface (API), written in Python (compatible with Python 2.7-3.5) and capable of running on top of TensorFlow, CNTK or Theano (see [89] for the complete library description). This deep learning library allows for easy and fast prototyping thanks to some characteristics like user friendliness, modularity and extendibility. Moreover, it supports both convolutional and recurrent networks, as well as combinations of the two and it runs seamlessly on CPU and GPU.
In particular, in this work I used Keras with TensorFlow and Theano as backend. Keras's models are *pre-trained* deep learning models which can be used for prediction, feature extraction, and fine-tuning: the network weights and parameters were obtained from a pretraining on the imageNet dataset and they are loaded automatically during the creation of the model.

4.2.4 TensorFlow

In my project I used TensorFlow as backend for running Keras applications. Specifically, TensorFlow is an open source software library for numerical computation which uses *data flow graphs* (see [4] for the complete library description). Nodes in the graph represent mathematical operations, while the edges represent *tensors*, i.e. multidimensional data arrays, communicating between them.
Thanks to his flexible architecture, TensorFlow allows deploying computation to one or more CPUs or GPUs in a desktop, server, or mobile device with the same API. It can be used for an enormous variety of applications despite it was originally developed by the Google Brain Team. This team belongs to Google's Machine Intelligence organization whose purpose is conducting research on machine learning and deep neural networks.

4.2.5 Theano

The second backend I used to run my Keras applications is Theano, an open source Python library which allows for fast numerical computation both on CPUs and GPUs [90]. Theano was developed in the LISA lab and its name was inspired by Pythagoras' wife, who was a Greek mathematician [5].
This library is commonly used for all those applications which presents large amounts of data like the implementation of Deep Learning pipelines where complex mathematical expressions are repeated more and more times.
Theano is build by two main components: a computer algebra system (CAS) and an optimizing compiler. It can not be defined as a programming language because the user writes a Python program which builds expressions for Theano: the Python library is, in that sense, an interface to a compiler.
Operations are expressed in terms of tensor operations by an high-level description language while the compiler uses lots of tricks to optimize these symbolic expressions (trade, heuristics, arithmetic simplification, symbolic differentiation, auxiliary libraries, memory aliasing, GPUs and so on) to speed up the computation.
Theano can be compared to hand-crafted C implementations for speed and optimization and can also generate customized C code for several mathematical expressions.

[4] https://www.tensorflow.org/
[5] http://deeplearning.net/software/theano/introduction.html

4.3 Hactar Cluster Polito

For running my algorithms, it was necessary to use the HPC clusters provided by the Polytechnic University of Turin [6]. This is an academic computer center managed by the LABINF (Laboratorio Didattico di Informatica Avanzata) under the supervision of DAUIN (Department of Control and Computer Engineering) which provides computational resources for research activities but also for didactic ones (thesis, academic projects, courses and so on). In Figure 4.2 are summarized the characteristics of the two clusters of the center [7]. In particular, a cluster consists in a set of connected computers (usually through a fast local area network) whose workload in scheduled by a scheduler software.

The processes that can run on a cluster are called *batch processes* because their execution is not interactive, i.e. their execution can be postponed. A job consists in one or more of these processes and its execution depends on the scheduler software which inserts the job in a queue with a priority that depends on the required resources and other factors. The scheduler used by CASPAR is the `SLURM` scheduler while the one of HACTAR is the `Grid Engine` one.

For this thesis I used the HACTAR cluster because is the only one with a high priority queue for GPU computations (compute-1-14, 2 Cores on 1 node + 2 GPU slots on a node). In fact, for running the training algorithms of my deep learning models, it was almost always necessary the graphic accelerations in order to complete the jobs. In fact the maximum duration of a job is 10 days and GPUs run much faster than CPUs, also if their fastness was for long time overrated [91].

Every job must be submitted to the scheduler through the `qsub` command followed by the reference to a script file (format `.qsub`) which contains substantially a bash script with a set of information for the scheduler and a sequence of directives for the command line.

In my case, a series of jobs with variable parameters had to be submitted both to the CPUs and GPUs queues. Therefore, a Python script for the automatic generation of the jobs and of the correspondent informations was written. This kind of approach was particularly useful especially in the first part of the transfer learning phase in which I was tuning the parameters. Then, once chosen the most suitable parameters, the same script was applied to these restricted set of parameters for every dataset.

[6] http://hpc.polito.it/index.php
[7] http://hpc.polito.it/download.php

Casper

Casper technical specificaion:

Architecture	Linux Infiniband-DDR MIMD Distributed Shared-Memory Cluster
Node Interconnect	Infiniband DDR 20 Gb/s
Service Network Gigabit	Ethernet 1 Gb/s
CPU Model	2x Opteron 6276/6376 (Bulldozer) 2.3 GHz (turbo 3.0 GHz) 16 cores
Performance	4.360 TFLOPS
Power Consumption	7 kW
Computing Cores	544
Number of Nodes	17
Total RAM Memory	2.2 TB DDR3 REGISTERED ECC
OS	ROCKS Clusters 6.1
Scheduler	GridEngine 2011.11p1

Hactar

Hactar technical specification:

Architecture	Linux Infiniband-QDR MIMD Distributed Shared-Memory Cluster
Node Interconnect	Infiniband QDR 40 Gb/s
Service Network Gigabit	Ethernet 1 Gb/s
CPU Model	2x Xeon E5-2680 v3 2.50 GHz (turbo 3.3 GHz) 12 cores
GPU Node	2x Tesla K40 - 12 GB - 2880 cuda cores
Performance	9.7 TFLOPS
Computing Cores	360
Number of Nodes	15
Total RAM Memory	1.9 TB DDR4 REGISTERED ECC
OS	CentOS 6.6
Scheduler	GridEngine 2011.11

Storage

Storages technical specification:

Home Storage	140 TB on RAID 6, throughput near 200 MB/s
Lustre Storage	87 TB. throughput greater then 2.2 GB/s
Storage Interconnect	Ethernet 10 Gb/s

Figure 4.2: Summary of the characteristics of the two HPC clusters and of their storage system that can be find on their website.

4.4 Stain Normalization

In histopathology it is really common to treat tissue samples using *stains*, i.e. colouring agents which are reactive only to specific biological substances. Therefore, according to the biological substance that the clinician wants to retrieve and underline, different stains are used. For example, using the H&E stain, hematoxylin underlines as blue-purple the nuclei acids while eosin underlines as bright pink the proteins [92].
Generally, stains absorb the light that the microscope points on the sample from to generate the image [92]. Thus, the areas reactive to a given stain will absorb light, while in the areas that are not reactive the light passes through giving them a bright white appearance.
The so-called *stain vector* contains the proportion of each wavelength absorbed [92]. Considering images acquired using an RGB camera, as for the Camelyon16 and GlaS dataset, the wavelengths of interest are only three: the red, the green and the blue one. Therefore, the stain vectors contains three elements. According to this reasoning, the amount of absorbed light and consecutively the absolute color values of the obtained slide must be proportional to the amount of the substance of interest.
Unfortunately, the amount of biological substance of interest is not the only cause of color variance among slides [92]. The absorbed light for a given unit of stain is really variable among different stains but also for the same one according to the producer, the time and method of storage (it must be noted that stains are sensible to light, thus they can fade if exposed to it), how much and how the stain is applied to the sample and many other factors. So this quantity can vary also between different slides prepared in the same lab as it can be seen in Figure 4.3.

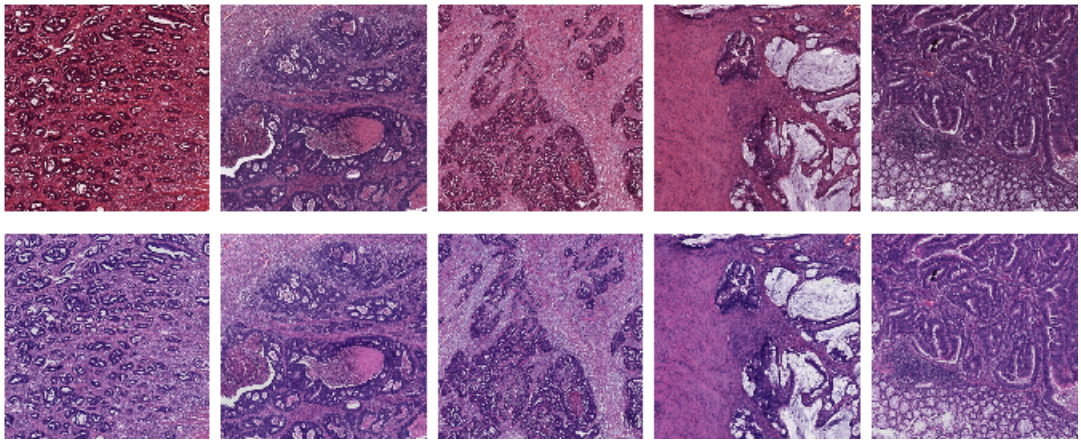


FIGURE 4.3: (First row) colorectal cancer H&E stained histopathology images, (second row) the same images after stain normalization [93].

This variability is addressed by clinicians thanks to their experience but it can be an issue for automatic algorithms for which drastically reduce the performances [93]. One possible solution proposed in the last years is the *stain normalization* (SN), which has shown promising results in many applications and also in CNNs-based algorithms for H&E images processing [93]. In fact, the authors of [93] compared two of the main state of the art SN techniques, the one of Bejnordi et al. [78] and the one of Macenko et al. [92] and showed through experimental results the importance of the presence of SN in the training pipeline of a CNN for histopathological purpose. In general, the aim of an SN algorithm is to match stain colors of a given image with a given template in order to normalize the dataset.

Personally I chose for my pipeline the SN method proposed by Macenko et al. [92] because it was the TUPAC and AMIDA reference SN method. It is a fully automatic algorithm which requires few parameters and no optimization and learning for finding the right stain vectors for an image and then performing the color deconvolution. The implementation I chose was provided by the `deep-histopath` Python project on GitHub [8] developed for the TUPAC16 challenge within the MICCAI 2016 conference, whose goal was the automatic prediction of tumor proliferation scores from WSIs of breast tumors.

First of all the color values must be converted into the correspondent optical density (OD) values [92]. This is done by taking the RGB color vector I, normalizing its components between $[0, 1]$ and then applying the following operation [92]:

$$OD = -log_{10}(I) \tag{4.1}$$

This operation creates a new color space OD where the color of each pixel can be expressed as a linear combination of two vectors corresponding to the two stains present in the image [92]. This is based on the assumption that exist two specific stain vectors corresponding to each of the two stains present in the image [92].

At this point, data with an OD intensity less than a given β, i.e. transparent pixels, are removed. This is done for every color channel, rather than considering the average over all channels for a given pixel. Experiments showed that the optimal value for β is $\beta = 0.15$ [92].
Then the eigenvectors of OD are computed and the two largest eigenvectors are extracted. Now the optical density values are projected on the plane generated by these two eigenvectors and the angle that each point generates with the first plane direction is computed. Then the maximum and minimum of these angles are evaluated (for a robust measure the α and $100 - \alpha$ percentiles are considered) and converted back to optimal stains in OD space.
Now that both the stain (V) and OD vectors are determined, the saturation of each of the stains are computed as the following color deconvolution:

$$OD = VS \Rightarrow S = V^{-1}OD \tag{4.2}$$

[8] https://github.com/CODAIT/deep-histopath

4.5 Training of UNet on Camelyon16

4.5.1 Why Camelyon2016

The first element to choose when implementing a transfer learning pipeline is the dataset that will be used for training the deep model. This is a really important choice because the network will learn relevant features of this dataset to perform his segmentation task and then the same features will be used for a completely new tasks.
As seen in the Section 3.2.2, hystopathological datasets are quite difficult to collect with respect to more common datasets like `Imagnet` or `MNIST`. They are really expensive to collect, usually not publicly available and the ground truth is really hard and time-consuming to annotate. Thus, it is really difficult to find large datasets in this field.
These considerations led me to choose the Camelyon2016 as my training dataset because of its interesting characteristics. First of all it was publicly available for the Camelyon16 challenge, second it was fully annotated. Last but not least it contains 400 WSIs, i.e. high resolution images which can be tiled in order to obtain a thousand of cropped images. Moreover, it was collected in two different medical centers which led the dataset to have a considerable variability due to the different acquisition conditions. More information on this dataset can be found in Section 3.3.3.

4.5.2 Why UNet

The second element to choose was the deep learning model used for performing the semantic segmentation. Analyzing the pipelines of the teams that took part to the Camelyon16 challenge and considering only those of them that used deep learning models, it was evident that the large majority did not use networks developed for semantic segmentation. Most of the times they used network developed for a single patch classification as seen in Section 2.3.1: usually each patch is classified as cancerous or not cancerous and then all the predictions are embedded into a unique heatmap image which highlights the tumor area. This second phase usually is performed by a second traditional machine learning classifier. The first algorithms of the final ranking are described in Section 3.3.5.
Instead, I was searching for a deep learning architecture that could perform a segmentation task end-to-end, i.e. the output of the network is directly the segmentation mask without the need to embed any single classification outputs.
In this sense the Warwick-QU-team presented a really interesting framework from which I took inspiration: they used a UNet architecture which is able to output directly a segmentation mask starting from the raw image. Even if it was not top-ranked, the results are also really good. A complete description of their pipeline can be found in Section 3.3.5
The choice of such a kind of network is really important in a transfer learning view of the problem because is free from all those post-processing operations which are hand-crafted and hard coded for a specific task and dataset.

4.5.3 Pipeline

The pipeline followed for the training of UNet on Camelyon16 dataset is depicted in Figure 4.4. The two fundamental elements are, as said before, the Camelyon16 dataset and UNet. The result of this pipeline is *knowledge*, which in this particular transfer learning approach is represented by the weights of UNet after being trained on Camelyon16. These weights will be used in the next two phases to initialize the network before starting the fine-tuning.

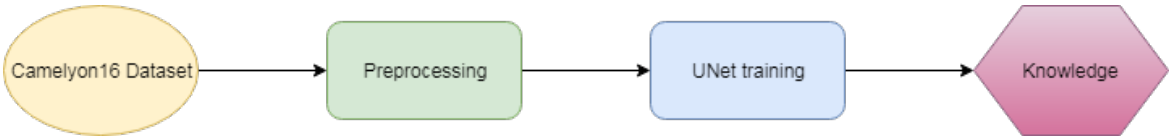


FIGURE 4.4: Pipeline followed for the training of UNet on Camelyon16 dataset

4.5.4 Preprocessing of Camelyon16

This section explains the **preprocessing** phase which every Whole Slide Image (WSI) of the Camelyon16 dataset is subjected to.

Since CNNs can not handle `.tif` input files directly, their input must be reshaped in the form of the so-called *patches*: small crops extracted from the original WSI with a fixed pixel size. The choice of this size as well as the number of patches will affect the performance of the CNN as well as its computational complexity. Therefore, the creation of the dataset represents a crucial step in working with CNNs and it requires lot of preprocessing of the original input.

The following section presents the preprocessing of a *single* WSI in order to get RGB patches of size 512×512 pixels at magnification $10\times$ of the original image. The patches are extracted only from the ROI corresponding to the tissue regions. The portion of the original image corresponding to the background and fat tissues are then a priori discarded. For this purpose an initial tissue segmentation is performed using Otsu's adaptive thresholding and some image-processing filtering. Subsequently, the patches are extracted only from the ROIs, stain-normalized and then discarded according to the amount of tissue in the patch.
Obviously in the real code this procedure was repeated for all WSIs.
The main libraries used for the preprocessing are:

1. `OpenSlide Python` which is a Python interface for `OpenSlide` library explained in Section 4.2.1 used for an easy managing of the **.tiff** format;

2. `Scikit-image`, a collection of Python algorithms for image processing used for the tissues segmentation and for the evaluation and filtering of the patches [9];

3. `deep-histopath`, which is the Python project developed for the TUPAC16 cited in Section 4.3 [10] that I used mostly for his stain normalization implementation.

4. `utils` module which is a collection of useful function that I implemented for an easy managing and preprocessing of the datasets.

[9] http://scikit-image.org/
[10] https://github.com/CODAIT/deep-histopath

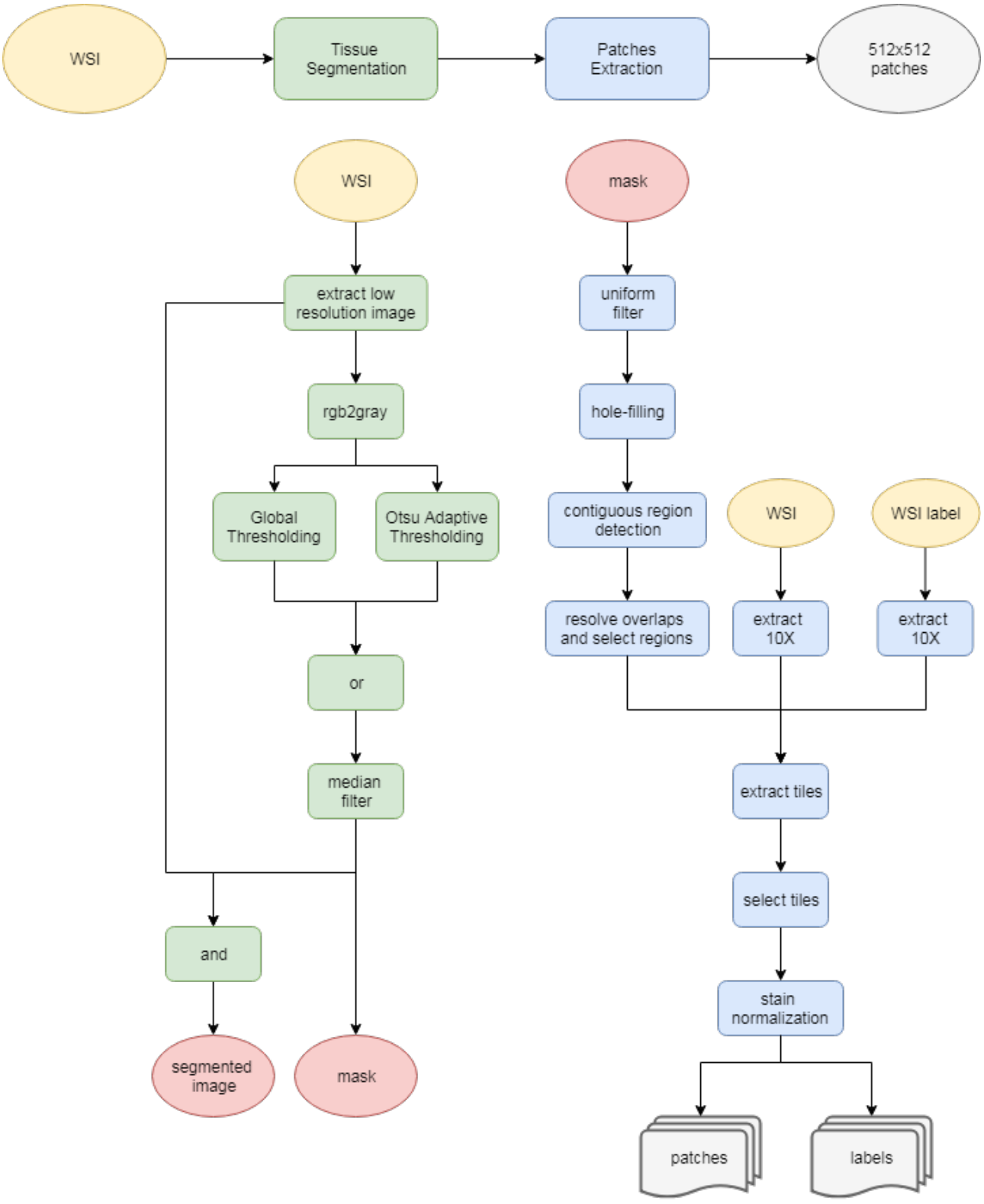


FIGURE 4.5: Pipeline followed in the preprocessing of Camelyon16 dataset.

Tissue segmentation

First of all the ROIs are segmented in order to reduce the amount of data to be considered. In fact, looking at the WSIs of the Camelyon16 Dataset (example in Figure 4.6 (A)), we can notice large blank regions corresponding to the background and large pale portion without cell tissue corresponding to fat: both of them should be discarded.

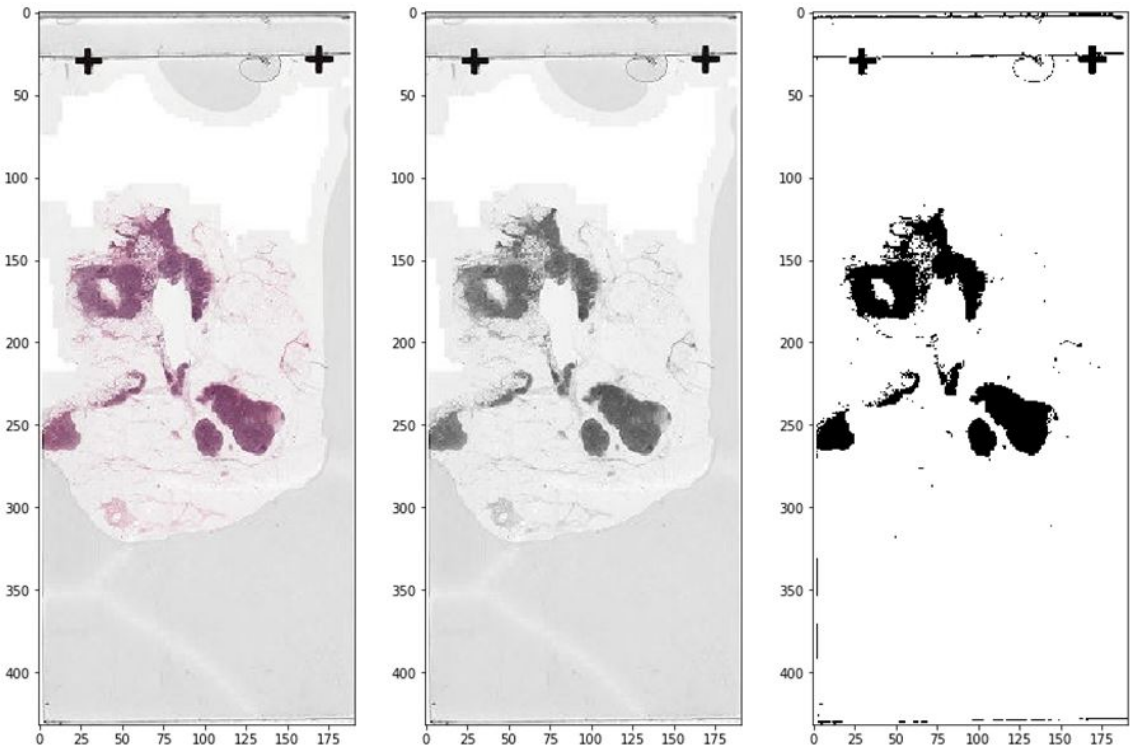


FIGURE 4.6: (A) RGB image extracted from the WSI, (B) grayscale version, (C) image after Otsu's adaptive thresholding.

In practice, this first phase consists in a *raw* histogram-based segmentation. It can be considered raw because it is perform directly on the lowest resolution of the `.tif` file and then projected on the full-resolution image. This allows to avoid demanding computational resources. The size and the magnitude of the tile which will be segmented is chosen in order to get a single tile containing the whole image.

Therefore, the ROIs and the correspondent masks are extracted at the chosen magnitude level from the WSI using `OpenSlide` routines. The extracted image is then transformed from the RGB format to the grayscale one in order to apply Otsu's adaptive thresholding (see Figure 4.6).
Otsu's method is particularly successful and famous for its simplicity and effectiveness on all those applications were the histogram of the images of interest can be considered bimodal, i.e. there are two different and well distinguished pixel distributions, one related to the ROI while the other one related to the background. In that sense, Otsu's method is able to compute the optimal threshold that maximize the variance between these two classes of pixels and minimize the intra-class variance of each of the two classes [11].

[11] http://scikit-image.org/docs/dev/auto_examples/xx_applications/plot_thresholding.html

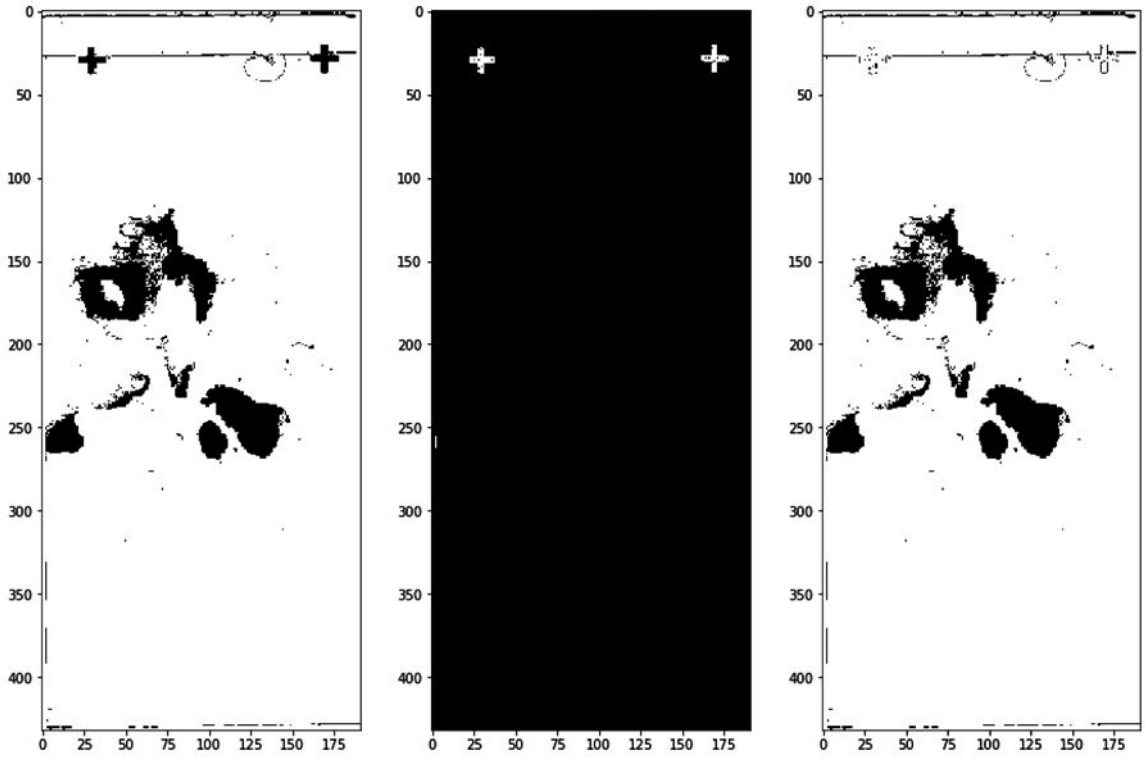


FIGURE 4.7: (A) Binary mask after Otsu's thresholding, (B) mask after hand-fixed global thresholding, (C) final or-merged mask.

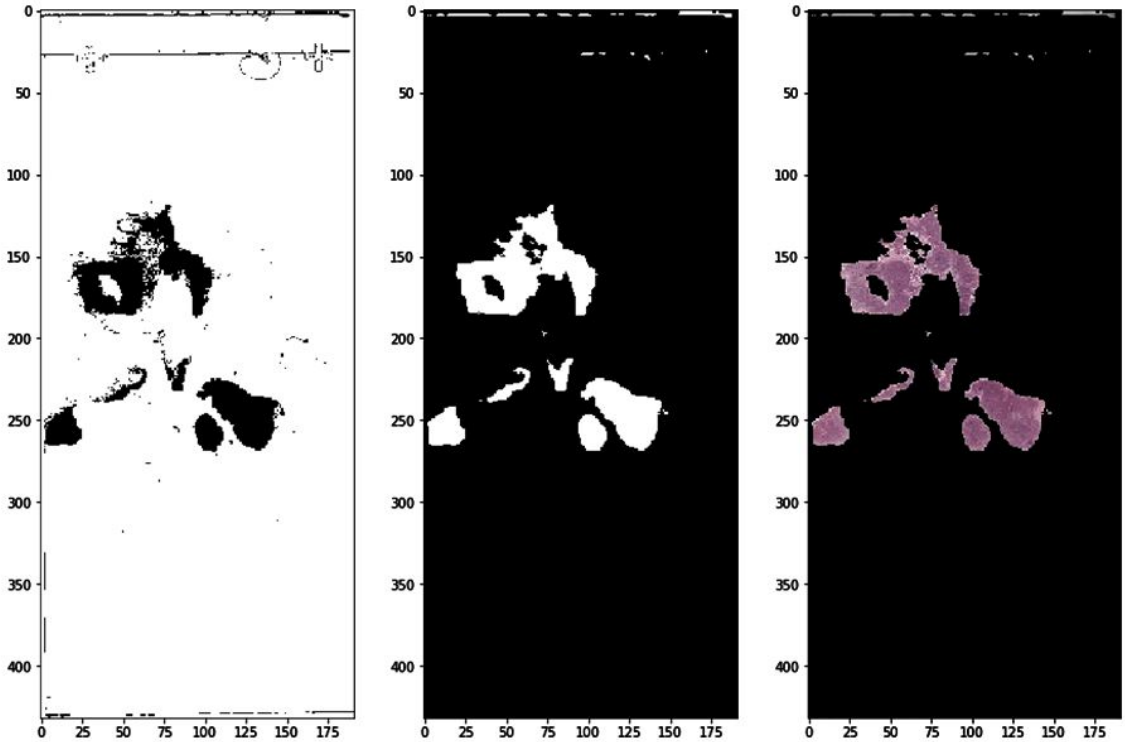


FIGURE 4.8: (A) Final mask, (B) complementary of the final mask, (C) segmented image.

Thanks to Otsu's thresholding I could separate the tissue regions from the background. In this case the tissue represented a dark object on a light background. Therefore, Otsu's thresholding could not remove the holding pins, i.e. those really dark objects present in most of the Camelyon16 slides. I removed these object through a simple fixed global thresholding, with an hand-tuned threshold. Then the obtained mask is merged with the one obtained through Otsu's thresholding through an `or` operation. The merged mask can be seen in Figure 4.7.

The final mask is obtained after applying a median filtering (disk radius equal to 2). The final segmented image is obtained by a pixel-wise `and` between the original image and the complementary of the mask as can be seen in Figure 4.8.

Tiles extraction

Now it is time to extract tiles from the ROIs obtained in the previous phase. First of all the most suitable zoom level for the tile extraction must be chosen. This parameter can be obtained knowing the maximal resolution of the WSI and the desired resolution for the patches we want to extract.

Often the maximal resolution of a `.tif` file is saved in his metadata but in our case we already know that it is 40×. We know that the desired magnitude is 10×, thus we can obtain the down-sampling factor between the magnification of the slide and the desired one as:

```
max_magnitude/desired_magnitude = 40/10 = 4
```

Now we can extract the zoom level offset from the highest resolution level, based on a 2× down-sampling factor in the generator as:

```
(max_magnitude/desired_magnitude)/2 = (40/10)/2 = 2
```

In our case the number of zoom levels is 19, i.e. from 0 to 18 because of the zero-based indexing, where 0 corresponds to the lowest resolution and 18 to the highest one. Therefore the desired zoom level for getting a 10× resolution in a slide with maximum 40× resolution is computed as:

```
highest_zoom_level - offset = 18 - 2 = 16
```

Extracting patches only from the ROIs obtained by the tissue segmentation is a critical problem because these ROIs are not regular. Therefore, for each image, the minimal rectangular regions that can contain the ROIs is computed. These rectangles are then pruned for their shape and merged in case of overlapping.

For the detection of contiguous regions a little preprocessing is required as can be seen in Figure 4.9. The input data are blurred so the ROIs have a continuous aspect through an uniform filter with smooth radius equal to 2 followed by an hole-filling to get cleaner regions.

At this point, the contiguous regions in the input array are detected and isolated with rectangle boundary boxes. As it can be seen in Figure 4.10 (A), lots of these boxes overlap. These situations are solved by replacing two overlapping boxes with the minimal box that contains both the contiguous regions as can be seen in Figure 4.10 (B). Finally, the boxes are pruned according to their area and eccentricity: very oblong boxes or very small ones are likely to not contain tissue regions. The result of this final step can be seen in Figure 4.10 (C).

At this point, using the `DeepZoomGenerator` function of `OpenSlide`, all the possible addresses of tiles of a given size with no overlap are extracted from the full

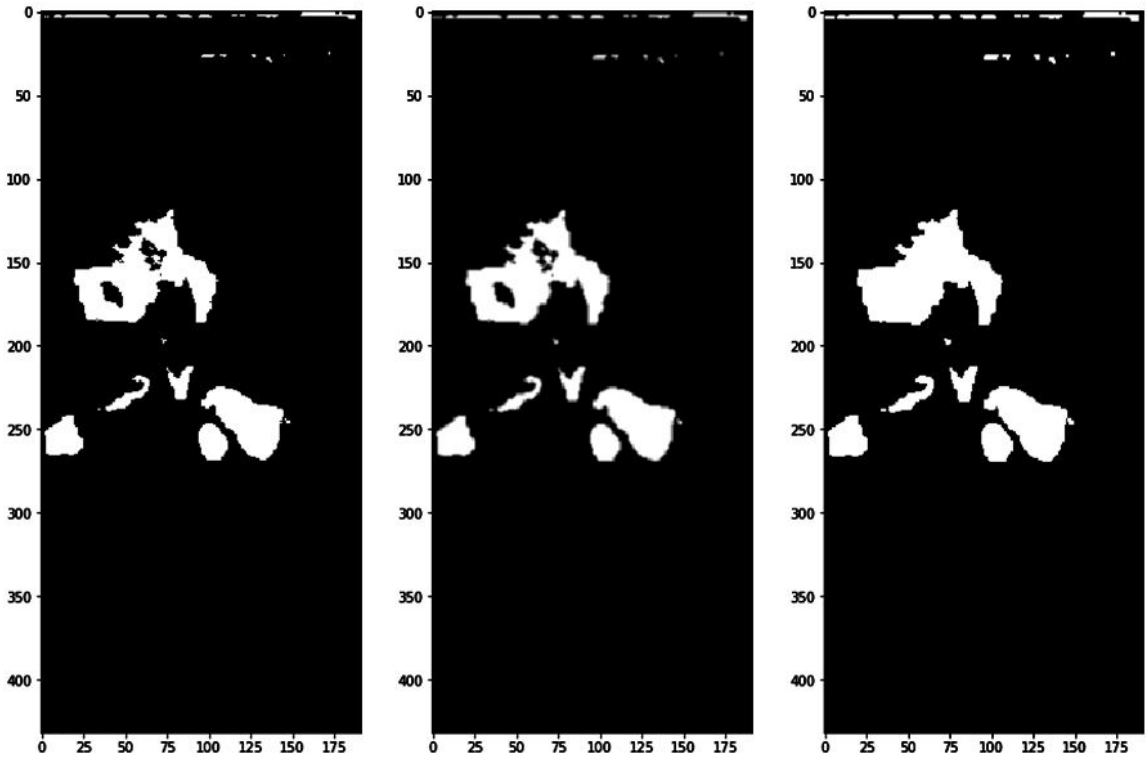


FIGURE 4.9: (A) Original mask, (B) mask after uniform filtering, (C) mask after hole-filling.

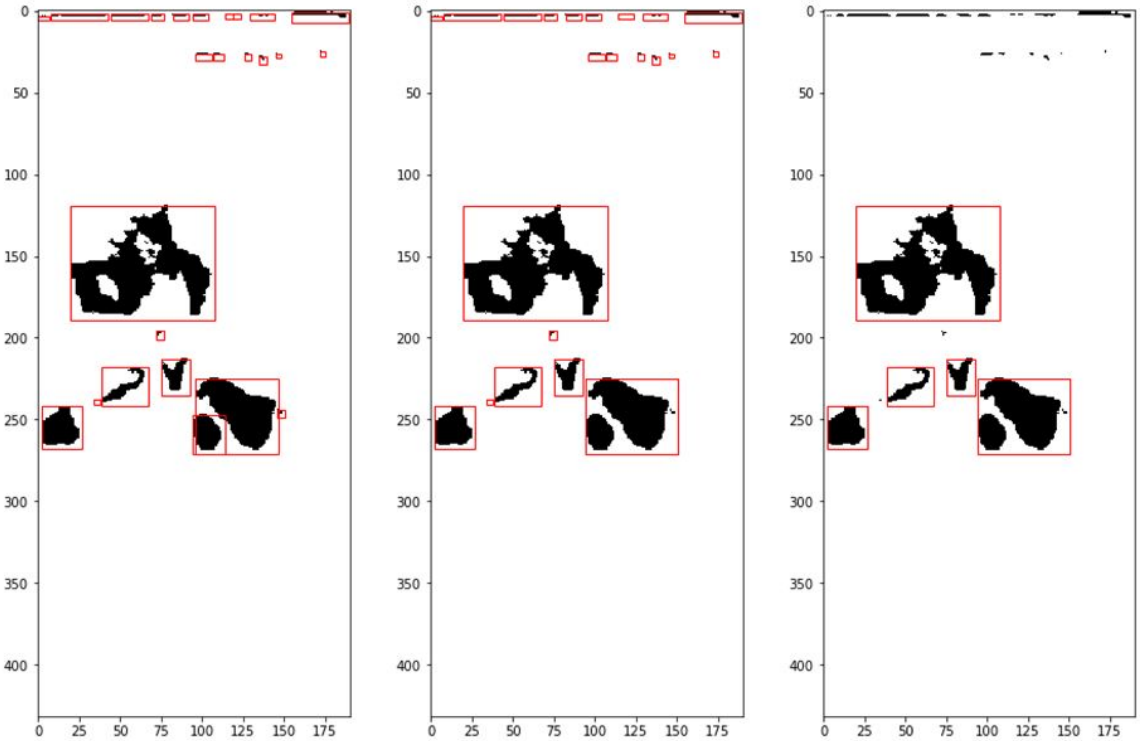


FIGURE 4.10: (A) Boxes containing tissue regions, (B) boxes after solving overlaps, (C) boxes after pruning too small and oblong boxes.

resolution image. This operation is repeated in an identical manner for the masks only for the metastatic WSIs of the Training Set while for all the other cases the masks are automatically generated as black images.
Now, only those addresses which are contained in the selected box at the desired zoom level are selected and their correspondent tiles are actually extracted. Only those tiles whose dimensions match and which contain more than the 90% of tissues are kept. The selected tiles are then normalized according to the H&E stain normalization described in Section 4.3 and then saved in the correct folder. These resulting tiles are 57048 and correspond to the entries of the dataset.

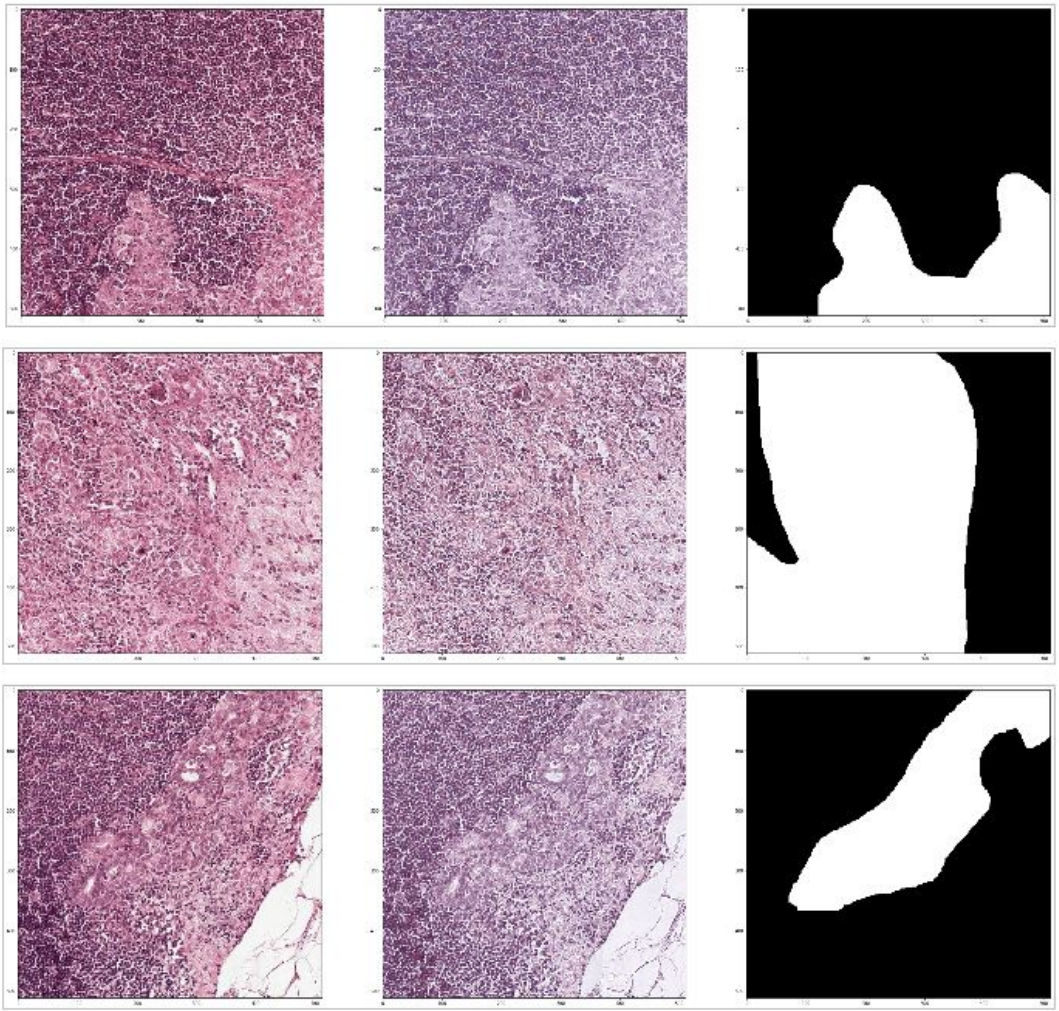


FIGURE 4.11: Examples of outputs of the preprocessing phase. Original tiles (left), their stain-normalized versions (middle) and their labels (right), i.e. the mask.

4.5.5 Training of UNet

In this phase, the UNet convolutional-deconvolutional neural network is trained using the Camelyon16 preprocessed tiles.

The number of tiles used for the training was reduce from 57048 to 12000 due to the excessive computational load required for training the entire dataset.

The outputs of the test are probability maps, whose pixel intensity values express the probability of being part of a cancer metastasis.

The main libraries used for the training are:

1. `Keras`, the Python neural networks API described in Section 4.2.3;
2. `Theano`, the open source library for fast numerical computation described in Section 4.2.5 which is used as first Keras backend;
3. `Tensorflow`, the open source library for fast numerical computation described in Section 4.2.4 which is used as second Keras backend;
4. `unet`, an UNet implementation with Keras for the TUPAC challenge provided by zhixuhao on GitHub [12].
5. `h5py`, a Python library for managing the HDF5 binary data format which allows for storing huge amounts of numerical data (e.g. deep learning models and weights) [13].

To train a `Keras` model, it is necessary to:

1. *define* the model;
2. *compile* the model;
3. *fit* the model.

Define the model There are two ways to define a `Keras` model: a *sequential* way through the `Sequential` class and a *functional* way through the `Model` class [89].

I can say that the sequential API is more intuitive to use because it allows to build models layer by layer in a sequential way, i.e. each layer can be connected only to previous and the next ones. This is also the main drawback of the class because it is not possible to define model with multiple inputs or outputs or models that share layers [14]. Instead, the functional API is less intuitive but allows to connect each layer to any other one. This allows for building more complex CNNs and also UNet, where layers of the convolutional subnetwork are connected to layers of the deconvolutional one.

In my personal case it was not necessary to build the model because UNet is a well-known State of the Art CNN whose implementations can be found easily on the web. I used the one provided by zhixuhao on GitHub which was, for the previously explained reasons, built using the `Model` class.

[12] https://github.com/zhixuhao/unet
[13] https://www.h5py.org/
[14] https://jovianlin.io/keras-models-sequential-vs-functional/

Compile the model UNet was then compiled in order to configure the model for training. It is a necessary step both before training and when loading pretrained weights [15]. In fact, the `compile` function of `Keras` takes the sequence of layers as input and transforms them into a new representation made of a series of matrix transforms. This new representation is expected to be more efficient and executable both on CPUs and GPUs.
The `compile` function of `Keras` requires many parameters. The most important are [89]:

- the *optimization algorithm* for training the network;
- the *loss function* (also known as *objective function*) that the optimization algorithm must minimize;
- the *metrics* to be evaluated during both training and testing.

There are lots of possible optimizer algorithms proposed by `Keras` like Stochastic Gradient Descent (SGD), RMSprop, Adagrad, Adadelta, Adam and so on [89]. Each of them has his advantages and drawbacks. In my implementation I chose the Adam optimizer, a first-order gradient descent optimization algorithm which often outperforms other methods as can be seen in Figure 4.12 for the MNIST dataset [94].

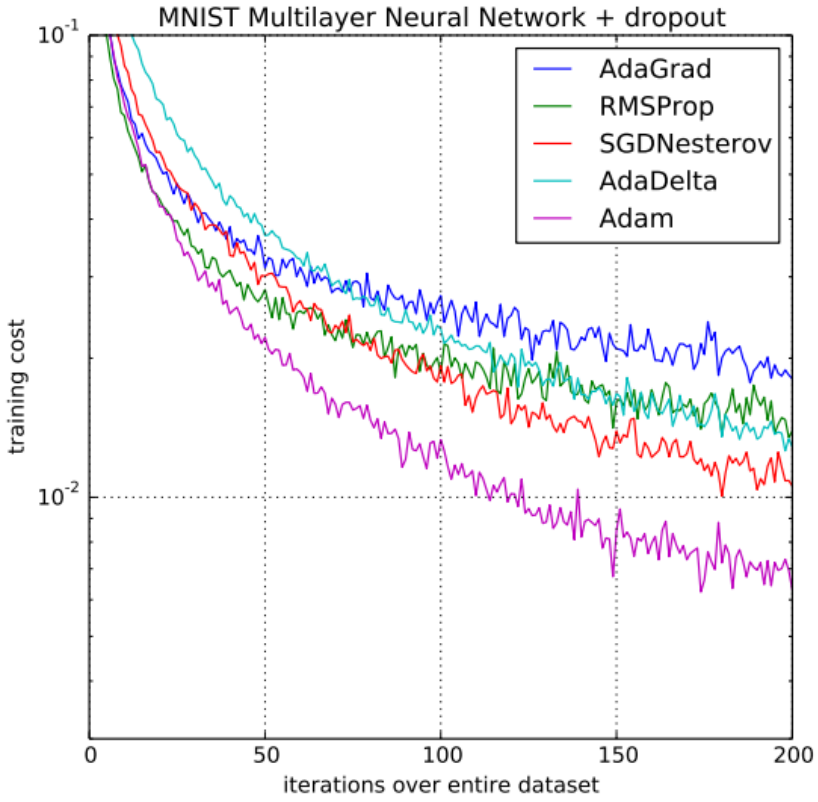


FIGURE 4.12: Cost function vs. iterations recorded during the training of a multilayer neural networks using dropout stochastic regularization on the MNIST dataset extracted from [94]. Different optimizers are compared in terms of convergence.

[15] https://machinelearningmastery.com/5-step-life-cycle-neural-network-models-keras/

This method adaptively estimates lower-order moments in order to minimize stochastic objective functions [94]. This means that the optimizer estimates the first and second moments of the gradients, i.e. the mean and variance, in order to compute individual adaptive learning rates for each parameter [16]. The only parameter I set for the Adam optimizer was the initial *learning rate* (*lr*).

The choice of the *loss function* is strictly connected to the prediction task [17]. For example, for a regression problem the best choice would be the mean square error while for a classification problem would be a logarithmic loss. In my implementation the *binary classification* loss function was chosen because the segmentation task requires to distinguish two classes: the region and the background one. In the case of a multiclass classification, i.e. if the classes were more than two, the best choice would have been the categorical crossentropy.

Fit the model Finally the model is trained using the `fit` function of `Keras`. The training parameters are summarized in the left Table of 4.1.
The epochs parameter is the number of epochs to train the model [89], where an epoch represents an iteration over the entire training data and label sets.
The batch size, instead, represents the number of samples per gradient update [89]. This parameter was chosen really small because a bigger one would have led to memory errors.
The validation split stands for the fraction of the training set that will be used as validation set [89], i.e. the set of data not used for the training that are used to evaluate the model at the end of each epoch.

To reduce the computational time, an additional check is added during the `fit` phase through a so-called `callback` function of `Keras`. It is called `EarlyStopping` and is able to stop the training when a monitored quantity has stopped improving. Here *improving* means both decreasing or increasing according to the chosen quantity to be monitored: for example, if the monitored quantity is an accuracy, the term improving means increasing while in the case of a loss, the term means obviously decreasing. In order to assert that the chosen quantity has *stopped improving*, this function checks if no absolute minimum change (referred also as *minimum delta*) is seen for a number of epochs (referred as *patience*). In my case, the number of epochs was initially set to 20 but the number of epochs actually completed during the training is 17 due to the action of the `EarlyStopping` function. The parameters set for the Early Stop callback are summarized in the left Table of 4.2.

One other callback of `Keras` called `ReduceLROnPlateau` is added in order to improve the performances. In fact, there are evidences that in the so-called *plateau*, i.e. phases in which learning stagnate, models benefits from reducing the learning rate parameter. Also this callback, like the previous one, monitors an user-chosen quantity and, in case there is not an absolute change bigger than a *epsilon* for the patience number of epochs, the learning rate is updated by a given *factor* in the following way:

$$new\_lr = lr \times factor \tag{4.3}$$

Obviously factor must be less than 1 because the learning rate should be reduced. I wanted to reduce the learning rate by a factor of 10 so I set factor equal to 0.1. The lower bound of the learning rate is set to 0. The cooldown parameter, which

[16] https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/
[17] https://machinelearningmastery.com/5-step-life-cycle-neural-network-models-keras/

represents the number of epochs before the normal operation is resume after the learning rate has been reduced, is set to 0. By the way, a summary of the parameters set for this callback is listed in the right Table of 4.2.

Parameter	Value
batch size	5
epochs	20
validation split	0.1

Parameter	Value
optimizer	Adam
learning rate	1e-4
loss	binary crossentropy
metrics	accuracy

TABLE 4.1: Training parameters: on the left table the parameter given as input to the `fit` function and on the right table the ones given to the `optimizer` function.

Parameter	Value
monitored quantity	validation loss
patience	2
min delta	0.0001
mode	decreasing

Parameter	Value
monitored quantity	validation loss
patience	2
epsilon	0.0001
mode	decreasing
factor	0.1
cooldown	0
min lr	0

TABLE 4.2: Callbacks parameters: on the left table the parameter given as input to the `Early Stopping` callback and on the right table the ones given to the `ReduceLROnPlateau` callback.

4.5.6 Testing UNet on Camelyon16

In this section I am going to present the performance of my implementation of UNet trained and tested on the Camelyon16 dataset. The performance and the results of the approaches explained in the following sections will bi presented in the next chapter.
In Table 4.3 the performance of UNet are reported both for the training and validation sets. A more close insight on the performance is given by the Figures 4.13 and 4.14 where the accuracy and the loss are plotted as function of the epochs. Since there are two objective function, accuracy and loss, the curves does not show monotonic behaviors: sometime the optimizer prefer one over the other. From the Figures 4.15 and 4.16, the overall training time takes around 11 hours and an epoch takes around 39 minutes on average.

UNet performances		
	training	validation
accuracy	0.973772068	0.9714989885
loss	0.07525953972	0.07562051125

TABLE 4.3: Final results of the Camelyon16 training.

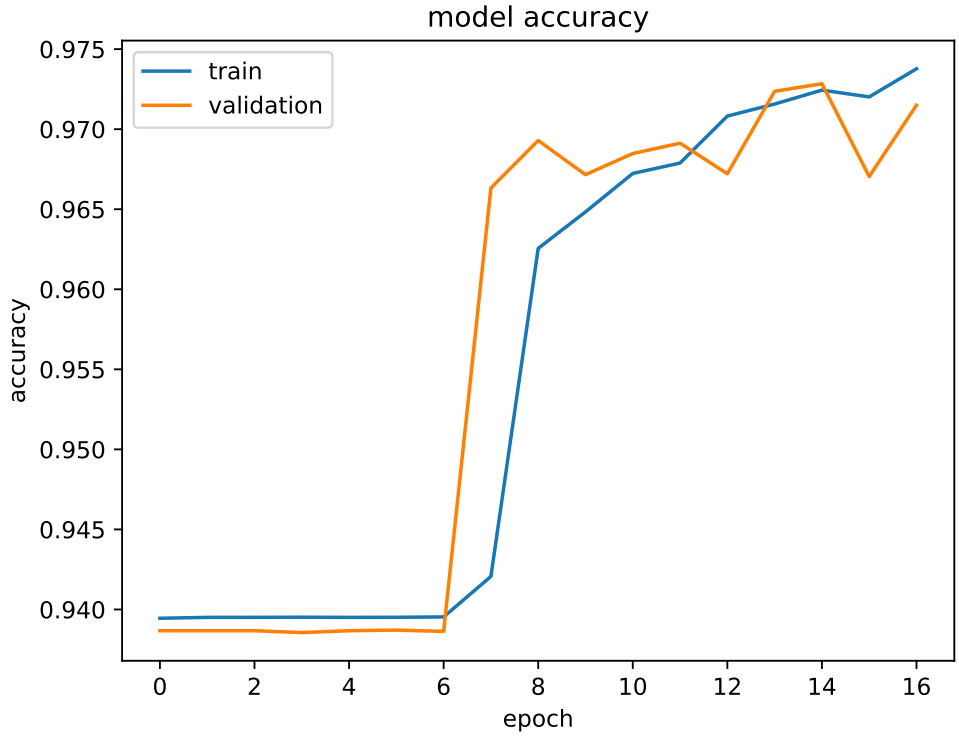


FIGURE 4.13: Training and validation accuracy of UNet trained on the Camelyon16 dataset

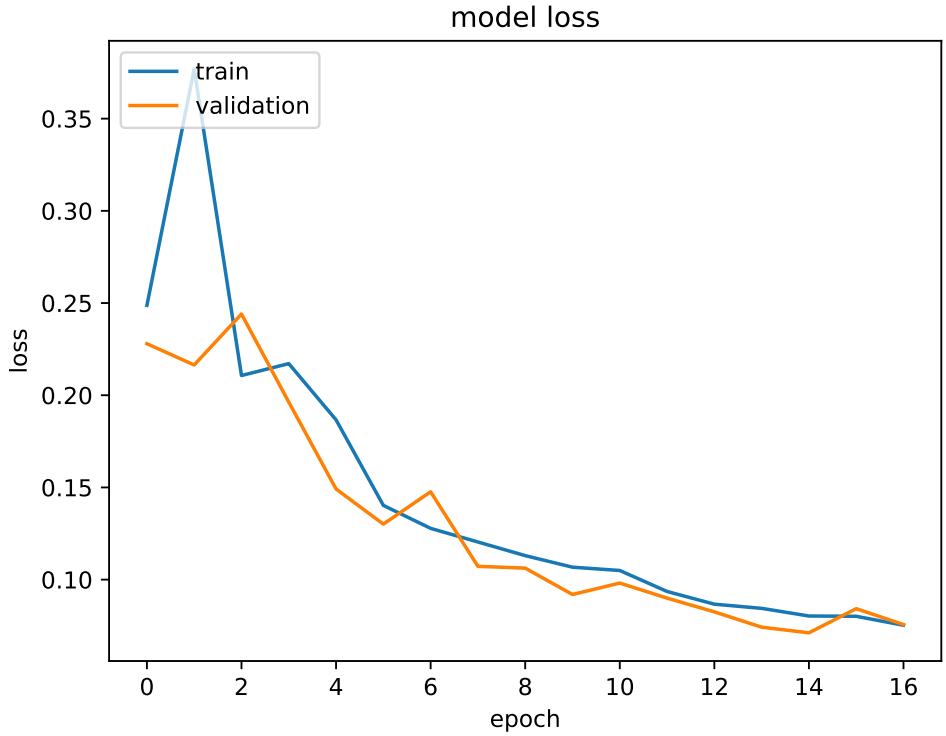


FIGURE 4.14: Training and validation loss of UNet trained on the Camelyon16 dataset

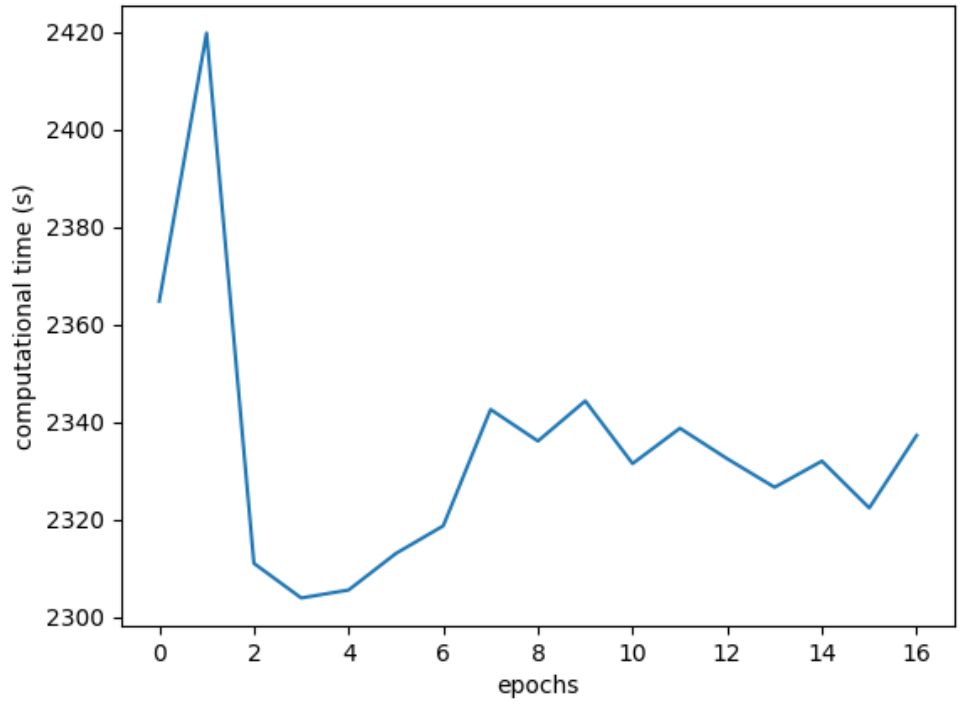


FIGURE 4.15: Computational time per epoch over epochs required for training UNet on the Camelyon16 dataset

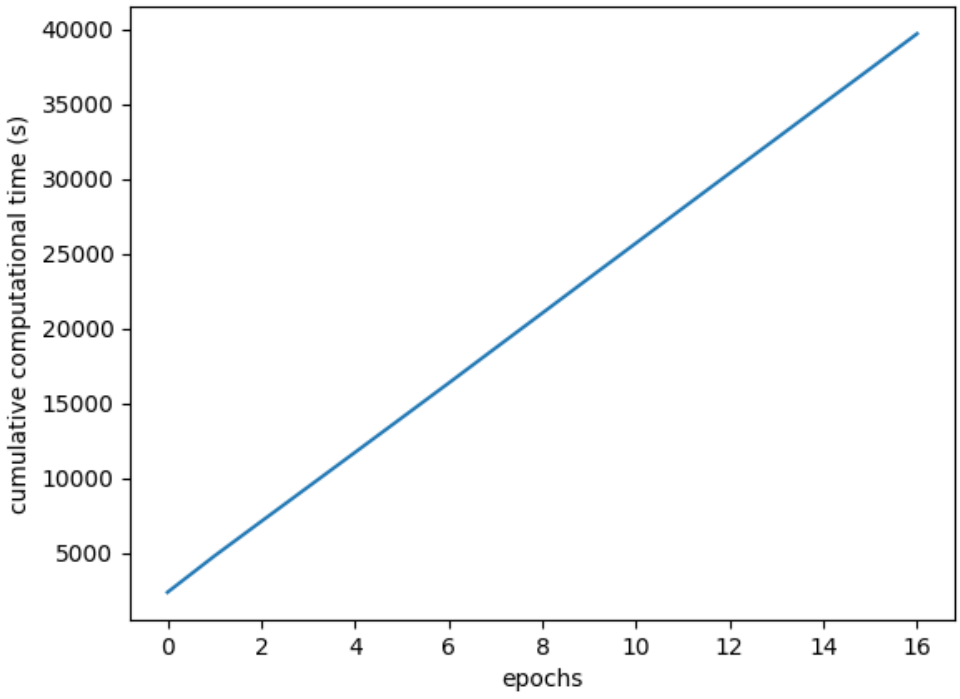


FIGURE 4.16: Cumulative computational time over epochs required for training UNet on the Camelyon16 dataset. The total time is almost 11 hours

4.6 Fine-tuning of UNet with augmented GlaS Data

4.6.1 Why GlaS Dataset

The GlaS Dataset was really interesting as second dataset for my transfer learning problem because it is a freely available dataset with a full-annotated ground truth. Moreover the main problem of this dataset is his size: in fact it contains only 165 images, 85 of them belonging to the training set and 80 belonging to the test set. Moreover these 165 images are cropped images taken from WSIs, therefore there is no possibility to extract more images by tiling as in the case of the Camelyon16 dataset.

One other interesting fact was that the content of this dataset is quite different from the Camelyon16 one because it contains 2D sections of glands. Thus, the observation scale is really different. The advantage is that also these images are H&E images, which may be an help for the network.
A more accurate description of the dataset is given in Section 3.4.

4.6.2 Pipeline

The pipeline followed for the Fine-tuning of UNet with augmented GlaS data is the one depicted in Figure 4.17. The GlaS dataset is first augmented and preprocessed and then given as input to the UNet model obtained with the training on the Camelyon16 dataset. The model is then fine-tuned in order to obtain a completely new model able to segment glands in histopathological H&E images.

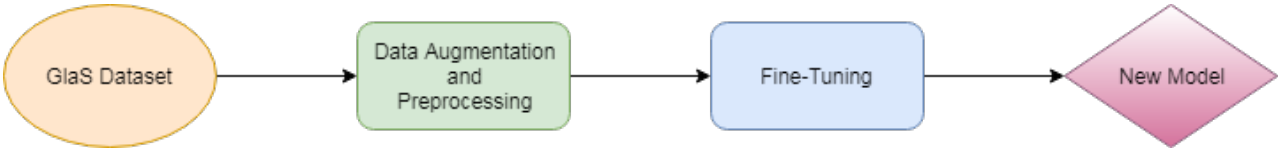


FIGURE 4.17: General pipeline for the Fine-tuning of UNet with augmented GlaS Data

Data Augmentation and Preprocessing

In this case data augmentation is necessary because the number of samples of the second dataset is so small that even the fine tuning will probably lead to bad results. So a data augmentation is performed in order to augment the number of training images for the fine tuning phase.

Operation	Probability	Additional parameters
random distortion	1	grid width = 4 grid height = 4 magnitude = 8
rotate	1	max left rotation = 5 max right rotation = 5
flip left right	0.5	
flip top bottom	0.5	

TABLE 4.4: Operations used for data augmentation of the GlaS dataset

The types of data augmentation are chosen based on biomedical considerations. For example, we can say that deformation is an image-processing operation that mirrors some biological aspects like the deformation of cells and tissues [10]. Mirroring and rotations are also possible operations because in histopathological images there is no prevalent direction like in the case of a portrait or a landscape.
Data augmentation is performed using the `Augmentor` library described in Section 4.2.2. The augmented dataset was sampled from the pool generated by the `Augmentor` generator function. In total, 1000 augmented tiles were sampled. All the operations used for performing data augmentation and their parameters are summarized in Table 4.4.
Then the augmented dataset is stain-normalized, resized (multiple factor 1 or 2) and cropped to 512×512 tiles.
The full pipeline of the preprocessing and data augmentation phase is depicted in Figure 4.18.

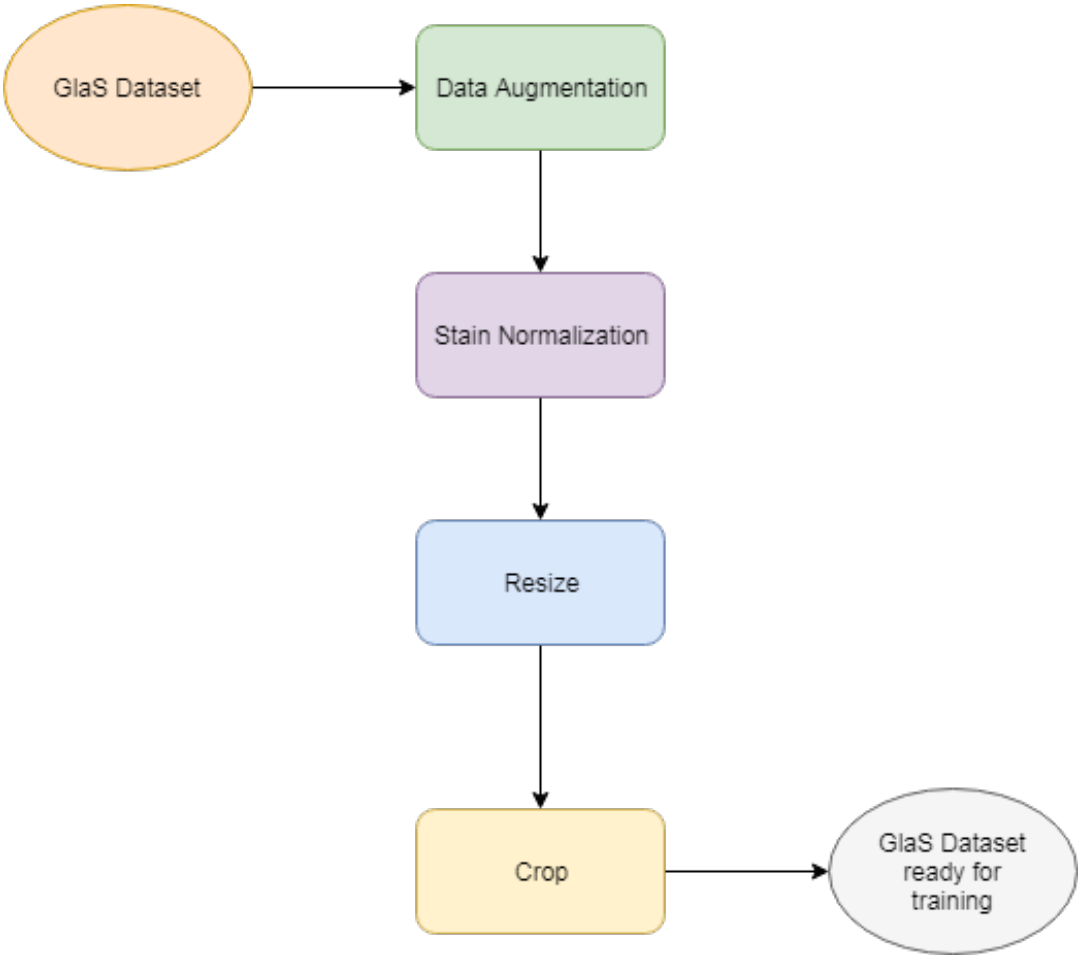


FIGURE 4.18: GlaS Dataset preprocessing pipeline

Fine Tuning

In the Fine-Tuning phase, the model trained on the Camelyon16 dataset is loaded and retrained for the new task. Therefore the training is not performed starting from a random initialization but from a precise set of weights.
Those weights can be selectively freeze in order to retrain only some layers. According to the idea that the features extracted at each layer level present and increasing

complexity and abstractness, probably the features extracted in the last layers are too specific for the original task while the one extracted in the first ones are generic and can be used for every task. Therefore is reasonably that the performance of the model on the new task will gradually decrease when the number of layers to freeze increase. On the other side, the more layers are retrained, the more the computational time and resources are required. So it is important to find the correct trade-off between performances and the computational load required for achieving them.
Thus, to evaluate the benefits of using a pre-trained model and find this trade-off, I trained the model varying the number of layers to freeze. The range of variability of this parameter is from 0 to 30 with steps of 10 layers. These values are chosen because the total number of layers of UNet (considering all the layers and not only the convolutional ones) is 39.
There is no need to modify the last fully connected layer. This is usually done when the number of classes changes, i.e. the network is trained to recognize a specific number of classes and then, when transferring the knowledge, the network has to recognize a different number of classes. In our case, the output is always a 512×512 black and white mask (two classes), therefore there is no need for modifying the last layer.

Parameter	Value
batch size	10
epochs	50
validation split	0.1

Parameter	Value
optimizer	Adam
learning rate	1e-4
loss	binary crossentropy
metrics	accuracy

TABLE 4.5: Training parameters for the Fine Tuning of UNet with the augmented GlaS dataset.

Parameter	Value
monitored quantity	validation loss
patience	2
min delta	0.0001
mode	decreasing

Parameter	Value
monitored quantity	validation loss
patience	2
epsilon	0.0001
mode	decreasing
factor	0.1
cooldown	0
min lr	0

TABLE 4.6: Callbacks parameters: on the left table the parameter given as input to the `Early Stopping` callback and on the right table the ones given to the `ReduceLROnPlateau` callback.

In conclusion, to fine-tune the `Keras` model, it is necessary to:

1. *load* the model trained on Camelyon16;
2. *freeze* the non-trainable layers;
3. *compile* the model;
4. *fit* the model with the GlaS data.

The training and callbacks parameters used for the fine-tuning phase are summarized in Table 4.5 and 4.6 and are quite the same of the ones of the original training.

The only difference is the batch size parameter which is increased from 5 to 10 thanks to the small amount of data used for the train and the epochs parameter which is increase to 50 for the same reason (a single epochs last drastically less with a small amount of data in the training set).

Hyperparameter	Values
data augmentation	True, False
stain normalization	True, False
resize	$\times 1$, $\times 2$
number of frozen layers	0, 10, 20, 30

TABLE 4.7: Hyperparameters for the Fine Tuning of UNet with the augmented GlaS dataset.

In addition to the training settings, I searched for the optimal set of hyperparameters that led me to the best results of the learning algorithm. Therefore I retrained the model as many times as the combination of the parameters listed in Table 5.2 making a sort of grid search.

4.7 Fine-tuning of UNet with augmented ISBI Data

4.7.1 Why ISBI12 Dataset

Due to the really good results of the Transfer Learning to the GlaS dataset, I chose to perform Transfer Learning on a third dataset, with much more different characteristics. This dataset, in fact, does not contain H&E images, and the segmentation task is really different. Now the task is to segment the neurons membranes therefore a sort of boundary segmentation instead of semantic segmentation. Moreover the number of images is really small (30 for the training and 30 for the test).

4.7.2 Pipeline

The pipeline followed for the Fine-tuning of UNet with augmented ISBI12 data is the one depicted in Figure 4.17. The ISBI12 dataset is first augmented and preprocessed and then given as input to the UNet model obtained with the training on the Camelyon16 dataset. The model is fine-tuned in order to obtain a new model.

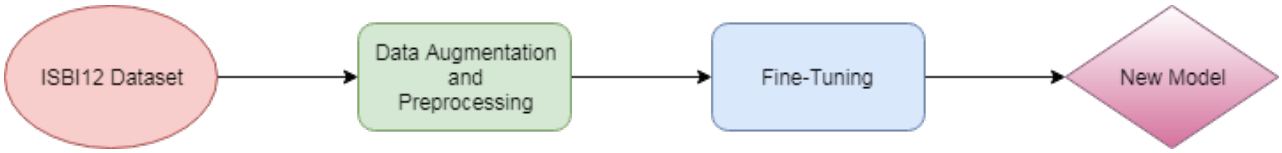


FIGURE 4.19: General pipeline for the Fine-tuning of UNet with augmented ISBI Data

Preprocessing and Data Augmentation

The preprocessing phase is really similar to the one followed for the GlaS dataset with the only exception that in this case no stain normalization is performed because the samples are ssTEM images and not H&E ones.
The full pipeline of the preprocessing and data augmentation phase is depicted in Figure 4.20. The main differences between this pipeline and the previous one is that here no stain normalization is performed because the dataset does not contain H&E stained images.

Uperation	Probability	Additional parameters
random distortion	1	grid width = 4 grid height = 4 magnitude = 8
rotate	1	max left rotation = 5 max right rotation = 5
flip left right	0.5	
flip top bottom	0.5	

TABLE 4.8: Operations used for data augmentation of the ISBI12 dataset

In this case data augmentation is even more necessary than in the case of the GlaS dataset because the training samples are only 30. The types of data augmentation are chosen making the same biomedical considerations made in the previous Section. Therefore, deformations, rotations and mirroring operations are performed and

1000 augmented tiles were sampled from the pool generated by the `Augmentor` generator function. The operations used for performing data augmentation and their parameters are summarized in Table 4.8.
Then the augmented dataset is resized (multiple factor 1 or 2) and cropped to 512 × 512 tiles.

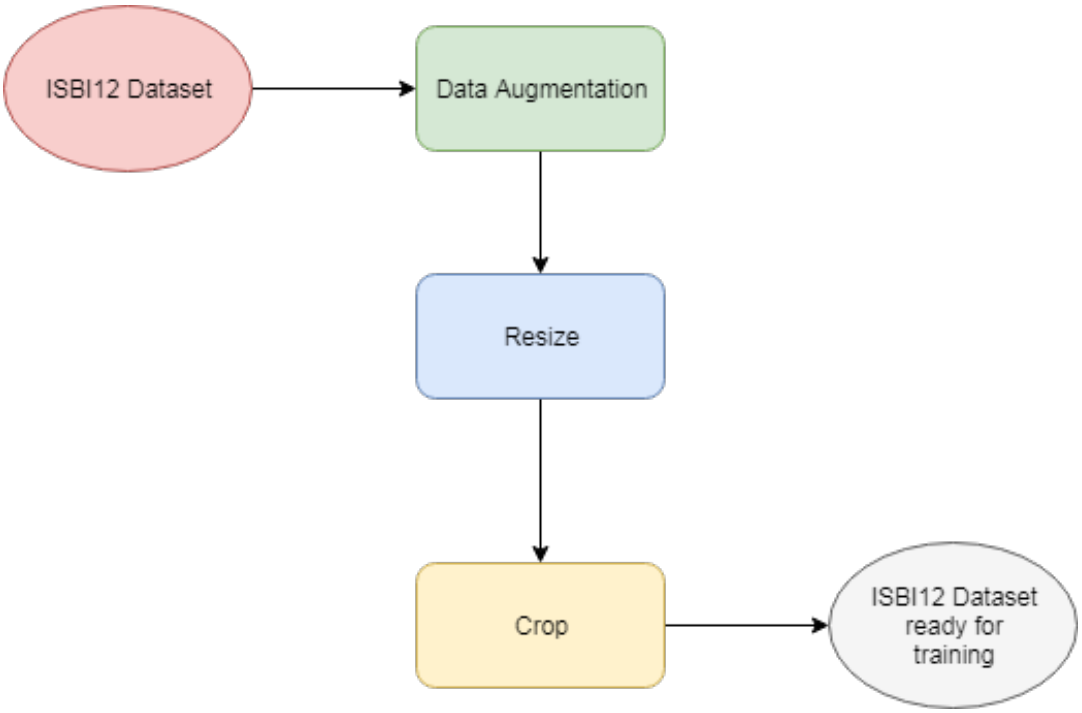


FIGURE 4.20: ISBI12 Dataset preprocessing pipeline

Fine Tuning

In the Fine-Tuning phase, following the same identical consideration made in the previous Section, the model trained on the Camelyon16 dataset is loaded and re-trained for the new task varying the number of layers to freeze. The range of variability of this parameter is from 0 to 39 with steps of 10 layers.
The performed steps are the following:

1. *load* the model trained on Camelyon16;
2. *freeze* the non-trainable layers;
3. *compile* the model;
4. *fit* the model with the ISBI data.

The parameters used for the fine-tuning phase are summarized in Table 4.9 and 4.10.

Also in this case I searched for the optimal set of hyperparameters that led me to the best results of the learning algorithm. Therefore I made a grid search using the parameters listed in Table 5.8. The main difference between the choice mode for the previous dataset are that here there is no stain normalization because of the nature of the data and no double resizing because it has shown to be bad choice with the previous dataset. Moreover, I tried also to freeze the total model with 39 frozen layers.

Parameter	Value
batch size	10
epochs	50
validation split	0.1

Parameter	Value
optimizer	Adam
learning rate	1e-4
loss	binary crossentropy
metrics	accuracy

TABLE 4.9: Training parameters for the Fine Tuning of UNet with the augmented ISBI12 dataset.

Parameter	Value
monitored quantity	validation loss
patience	2
min delta	0.0001
mode	decreasing

Parameter	Value
monitored quantity	validation loss
patience	2
epsilon	0.0001
mode	decreasing
factor	0.1
cooldown	0
min lr	0

TABLE 4.10: Callbacks parameters: on the left table the parameter given as input to the `Early Stopping` callback and on the right table the ones given to the `ReduceLROnPlateau` callback.

Hyperparameter	Values
data augmentation	True
stain normalization	False
resize	$\times 1$
number of frozen layers	0, 10, 20, 30

TABLE 4.11: Hyperparameters for the Fine Tuning of UNet with the augmented GlaS dataset.

Chapter 5

Results Analysis

In this chapter the results of the experiment are analyzed and a possible interpretation is given. In particular, the following analysis are performed:

- Learning Rate Analysis
- Hyperparameters Analysis
- Comparison with UNet trained on GlaS from scratch
- Comparison with UNet trained on ISBI12 from scratch
- Computational Analysis

Method An ad-hoc Python script automatically generates all the jobs required for the analysis. A single job corresponds to the fine-tuning of a specific model with the chosen hyperparameters. The results of each training and the corresponding metadata are stored in a database structure.

Computational Constrains A really important detail to remember is that a job can run on the HPC cluster for max 10 days before being killed by the scheduler. Moreover, the GPUs nodes are only two for the entire cluster and often busy. Therefore, due to the huge number of jobs and the time requested, I choose to the use mainly CPUs nodes. However this choice entails a slower speed in the job execution and some of them were not able to finish the training.

5.1 Evaluation Metrics

Evaluating the performances of a semantic segmentation algorithm is not so trivial as it seems. The evaluation can be performed in a multitude of ways which are, most of the times, strictly application-dependent and there are no standard evaluation metrics. This is obvious because a learning model is usually built to perform a very specific task, thus it is considered successful only in relation that task.

For example, as seen in Chapter 3, Camelyon16, GlaS and ISBI12 challenges propose really different metrics to evaluate their candidate models. All these metrics were closely related to the characteristics of the objects or regions of interest and to be segmented. For instance, there are low-level metrics, such as at pixel level, region-based metrics or contour-based metrics.

On one hand these metrics are becoming even more and more complex and difficult to be computed. This makes hard the comparison of State of the Art algorithms, especially for learning models. On the other hand, all the proposed approaches feature a strong use of post-processing steps which introduce an additional variability.

Instead I preferred to evaluate the benefits and highlight limitations only of the transfer learning and data augmentation approach. For that reason I performed no post-processing and the chosen metrics are chosen to be much more as possible application-independent.
I considered the segmentation problem as a pixel-level classification problem. At this level, it was not possible to evaluate the accuracy on the detected objects because of the intrinsic difference of the objects in the three datasets.

The metrics used for the evaluation of each model are the following:

- **Pixel-level Model Accuracy** (both for training and validation)
- **Pixel-level Model Logarithmic loss** (both for training and validation)
- **Computational Load** (in terms of time and memory)

Pixel-level Model Accuracy The Pixel-level Model Accuracy is the first and simplest metric based on the ground truth that was computed. It represents the ratio between true prediction of the classifier, which includes both true positives and true negatives, and the total number of cases examined [95].

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TP} + \text{FN}}, \tag{5.1}$$

where TP represents the number of true positives, TN the number of true negative, FP the number of false positive and FN the number of false negatives.
From one other point of view, the accuracy can be seen as the complement of the misclassification error which represents the percentage of pixels that are classified as belonging to one class when they belong to another class [95]:

$$\text{Misclassification Error} = 1 - \text{Accuracy}. \tag{5.2}$$

Other classification metrics that take into account the role of False Positive and False Negative, such as *Precision*, *Recall*, are not considered in this analysis.

Pixel-level Model Logarithmic loss A loss function, or cost function, indicates how well the built model fits the data [1]. This means evaluating how much the prediction differs from the actual value in an absolute way, i.e. how much incorrect they are no matter the direction. Therefore, if the loss is high, the predictions are totally wrong while if the loss is low the predictions are good. The loss function is also one of the necessary parameters required to compile a `Keras` model and to build some of his callbacks [2].
In the literature, many cost function have been proposed. The most famous loss functions are *Mean Squared Error*, *Mean Absolute Error*, *Likelihood loss*, *Logarithmic loss*.

Cross-entropy loss, or *Logarithmic loss* (log loss), measures the performance of a classification model whose output is a probability value, hence, a value between 0 and 1 [3] . In a binary classification task, the log loss increases as the predicted probability diverges from the actual label (0 or 1). So predicting a very low probability, e.g. 0.012, when the actual observation label is equal 1 would result a high loss value. A perfect model would have a log loss value of 0.

[1] https://blog.algorithmia.com/introduction-to-loss-functions/
[2] https://keras.io/losses/
[3] http://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html

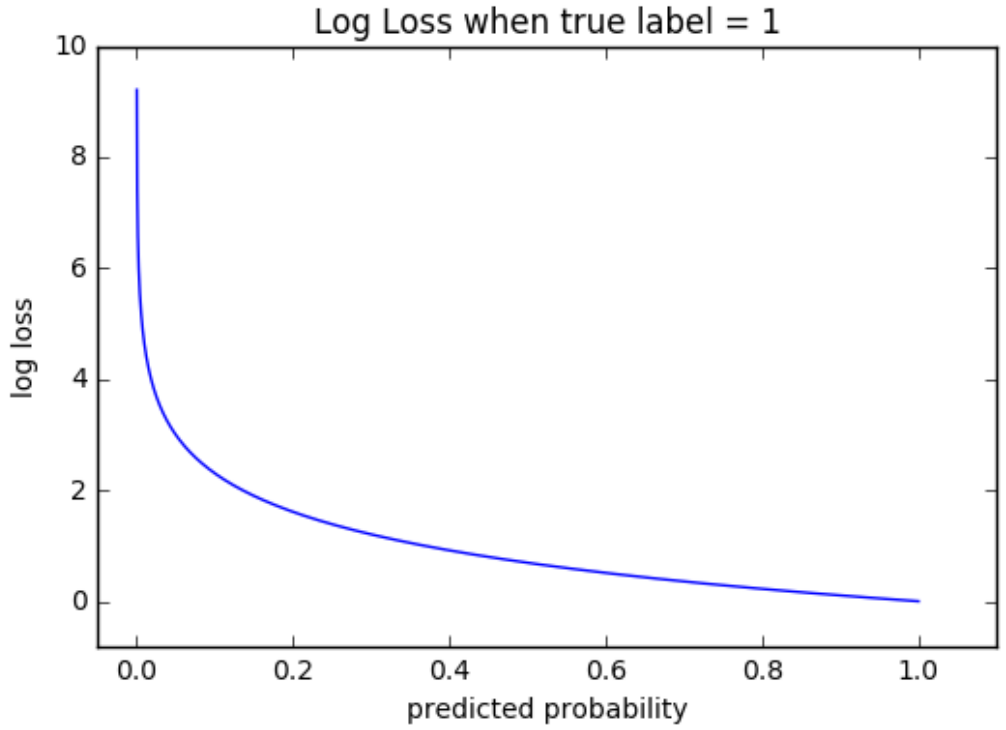


FIGURE 5.1: Logarithmic loss function considering.

In this thesis I choose to use the log loss. This really common loss function is just a logarithmic version of the likelihood function and his trend can be seen in Figure 5.1 (courtesy of [4]).

This function evaluates the cross-entropy value for a given binary classification problems as following [5]:

$$\text{logloss}(x, y) = -x \log(y) - (1 - x) \log(1 - y), \tag{5.3}$$

where log is the natural logarithm, x represents the ground truth label (0 or 1) and y represents the probability of the sample to belong to a certain class predicted by the model.

In Figure 5.1, it can be seen the range of possible loss values when a true observation is given[6]. In this case, if the model predicts a value near to 1 the loss is really low because the prediction makes only a little mistake. Instead, if the prediction is opposite, i.e. the model predicts class 0 when the ground truth is 1, the loss value explodes. This mechanism heavily penalizes when the model predicts the wrong class with an high confidence [7].

[4] https://blog.algorithmia.com/introduction-to-loss-functions/
[5] http://wiki.fast.ai/index.php/Log_loss
[6] http://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html
[7] https://blog.algorithmia.com/introduction-to-loss-functions/

5.2 Learning Rate Analysis

First of all a learning rate analysis is performed in order to understand which is the most suitable learning rate value for my particular task. The parameters setting for this analysis is shown in Figure 5.1. In particular, no data augmentation and no resize are done in order to speed-up the convergence of the model and see which learning rate is the more effective and efficient. The analysis is performed only on the GlaS dataset using Theano as backend on the CPUs nodes of the cluster.

Parameter	Values
learning rate range	$(10^{-1}, ..., 10^{-8})$
data augmentation	False
stain normalization	True
rescale	$\times 1$
number of frozen layers	20
dataset	GlaS
backend	Theano
cluster	CPU

TABLE 5.1: Settings for the Learning Rate Analysis

In the Figures 5.2 and 5.3, the training and validation performances per epoch are compared according to the learning rate. In the left column of both figures can be seen the model accuracy while in the right one the loss. The graphics in the first row of both figures are waterfall representation of the ones in the second row. The color scale goes from deep blue to light yellow which (yellow high accuracy (resp. loss) and blue low accuracy (resp. loss)). I am interested in those models with an high accuracy and a low loss. It is better if in a lower number of epochs.
From a first-view analysis, the most suitable learning rates are 10^{-3} and 10^{-4}. The performances of the model with a learning rate equal to 10^{-4} present a more suitable trend because the accuracy gradually increases while the loss gradually decreases. The performances of the model with a learning rate equal to 10^{-3}, instead, present a too fast trend which led me to guess it was a local minimum.
According to these considerations I chose as learning rate 10^{-4} for my models.

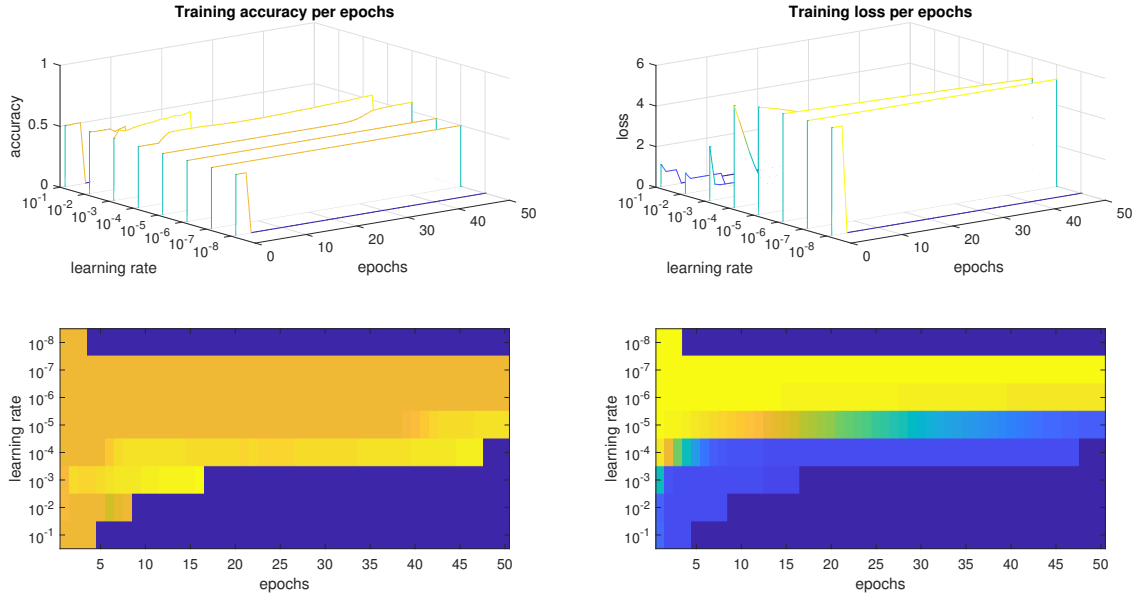


FIGURE 5.2: Training performances per epochs versus varying learning rate: model's accuracy (left column) and loss (right column).

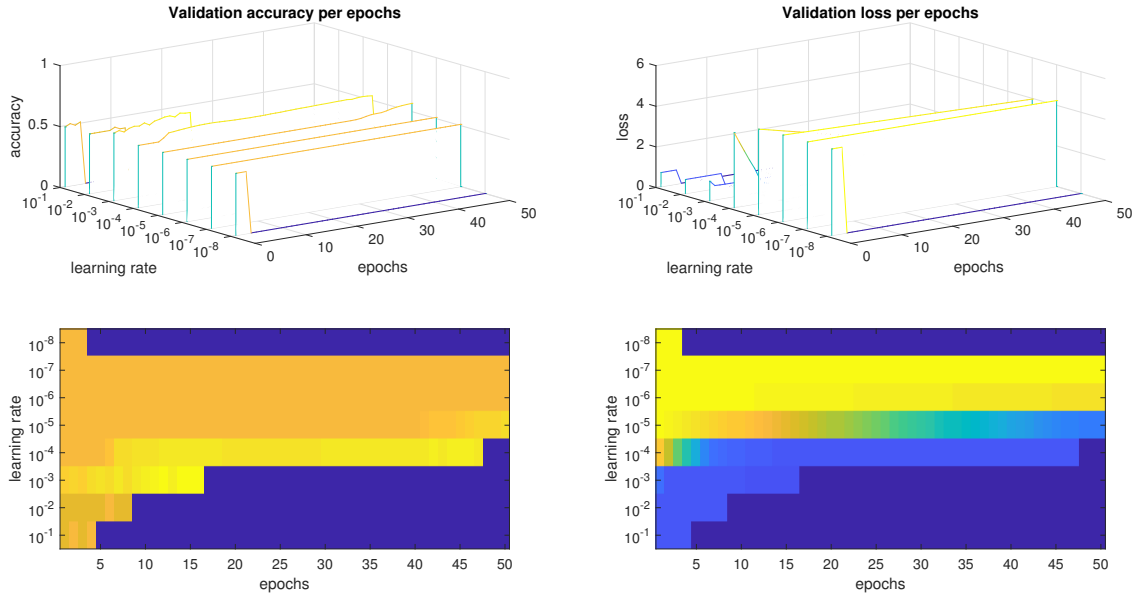


FIGURE 5.3: Validation performances per epochs versus varying learning rate: model's accuracy (left column) and loss (right column).

5.3 Hyperparameters Analysis

5.3.1 Glas Dataset

A grid search is performed in order to find the best set of hyperparameters for fine-tuning the GlaS dataset. For each combinations of hyperparameters a specific job is generated where a UNet model is fine-tuned with a given number of frozen layers and certain preprocessing steps.
In particular I wanted to evaluate if data augmentation provides some benefits on the performances and if stain normalization has some influence as stated in literature. Moreover I wanted to evaluate the influence of the different scale factor between the images of the new dataset with respect to those belonging to the Camelyon16 one. Finally the effect of depth of frozen layers, i.e. layers for which the original weights are frozen and not retrained, is analyzed.
The hyperparameters tested in this analysis are summarized in Table 5.2.

Parameter	Values
data augmentation	True, False
stain normalization	True, False
rescale	$\times 1$, $\times 2$
number of frozen layers	0, 10, 20, 30

TABLE 5.2: Hyperparameters for the Fine Tuning of UNet with GlaS data.

It is important to note that many models (10 models) were not able to complete the training during the 10 days time allowed by the HTC cluster, in particular the ones with the data augmentation preprocessing. These models are underlined in the graphics with a star.
Moreover, jobs which presented both data augmentation and the rescale preprocessing crashed due to memory error. These jobs are not considered in the graphics and in the final considerations. I did not even try to resume these jobs because one of the goal of my thesis is to reduce the computational resources required by the training of deep learning model, and these kind of models obviously do not match these requirements.
As final consideration, only 2 nodes with GPU were available, therefore all my 32 models ran in parallel on CPUs. Probably the same jobs would have finish the training if they ran on GPUs nodes.

The type of graphs used for the analysis are simple scatter plots where each model is located in a 2D plane according to his final validation accuracy (x-axis) and his final validation loss (y-axis). The model points are highlighted using different colors according to the hyperparameters used.

Data Augmentation

In Figure 5.4, the different model points are highlighted in different colors if data augmentation is performed or not. The red model points represents all those model for which data augmentation is performed while the violet ones all those models for which no data augmentation is performed. An additional two points are shown: one for the model trained from scratch using an augmented dataset (yellow) and without (green).

Considering only the final validation loss, it is evident that data augmentation provides big benefits: all the red points present a loss value minor then the violet

ones. This is quite logical because the loss value express how well the model fits a certain dataset and with data augmentation this dataset is hugely augmented. The loss is not improved, instead, with respect to the two reference models trained from scratch, both with and without data augmentation.

The consequence is not so immediate when talking about the accuracy. Only 5 models present accuracy values major than the 80% and all these models where trained using augmented data. A positive note is that no model had worst accuracies than the random classifier (no accuracies are under the 50%) and every model got a better accuracy with respect to the two reference models.

In conclusion, data augmentation drastically improves accuracy and reduces loss, therefore is a necessary operation of the preprocessing pipeline.

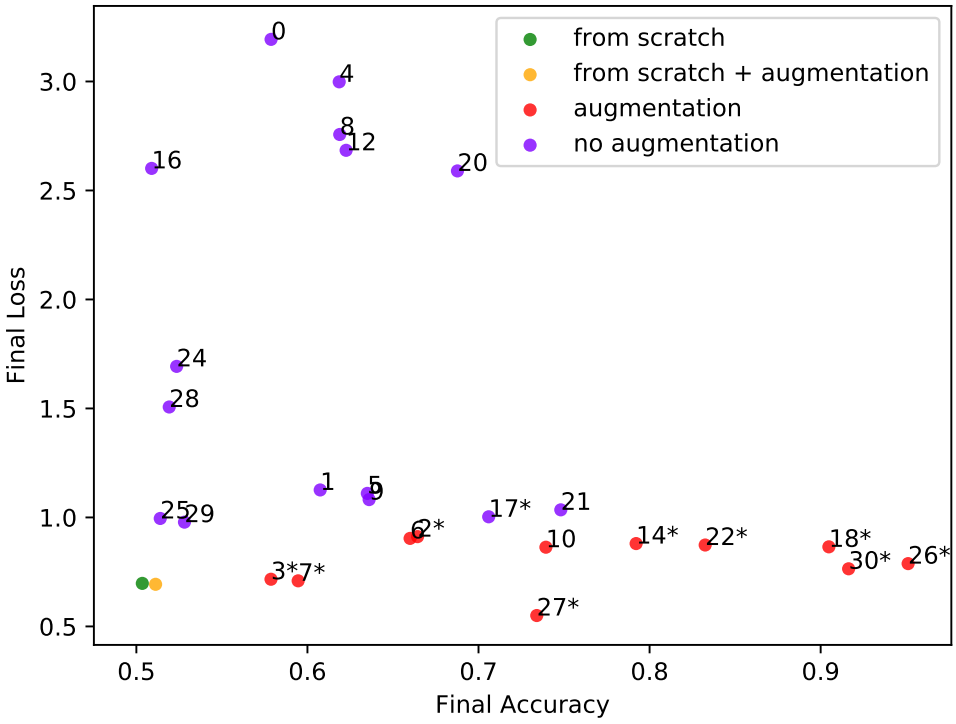


FIGURE 5.4: Augmentation vs. None (* = Interrupted job).

Stain Normalization

In literature, stain normalization is usually presented as a necessary preprocessing step when dealing with H&E stained images. This technique, described in Section 4.3, is widely used and can be implemented in a multitude of ways which can require significant computational resources.

It is interesting to evaluate if in my case such preprocessing is really necessary. In fact, even if my implementation of this technique is not computational demanding, this step adds an additional variability in the general pipeline which is likely to be avoided. Moreover the main goal of my approach is to avoid any unnecessary variability which represent an obstacle to the transfer of learning in more applications as possible. For example, if I would like to implement a standard tool for an easy

transfer learning in histopathology, I would have both *H&E* stained images and images stained using other colorants or even no colorants like in the case of the ISBI12 dataset.
Looking carefully at Figure 5.5, it can be seen that red and violet models are coupled. Red points represents all those models with a stain normalization preprocessing, while the violet ones presents no stain normalization. Looking at the other parameters, I discovered that each couple of models share the same parameters with the exception of the stain normalization. This means that the same model with a stain normalization preprocessing has quite the same performances of the one without stain normalization. Therefore stain normalization has no evident influence on the performances.

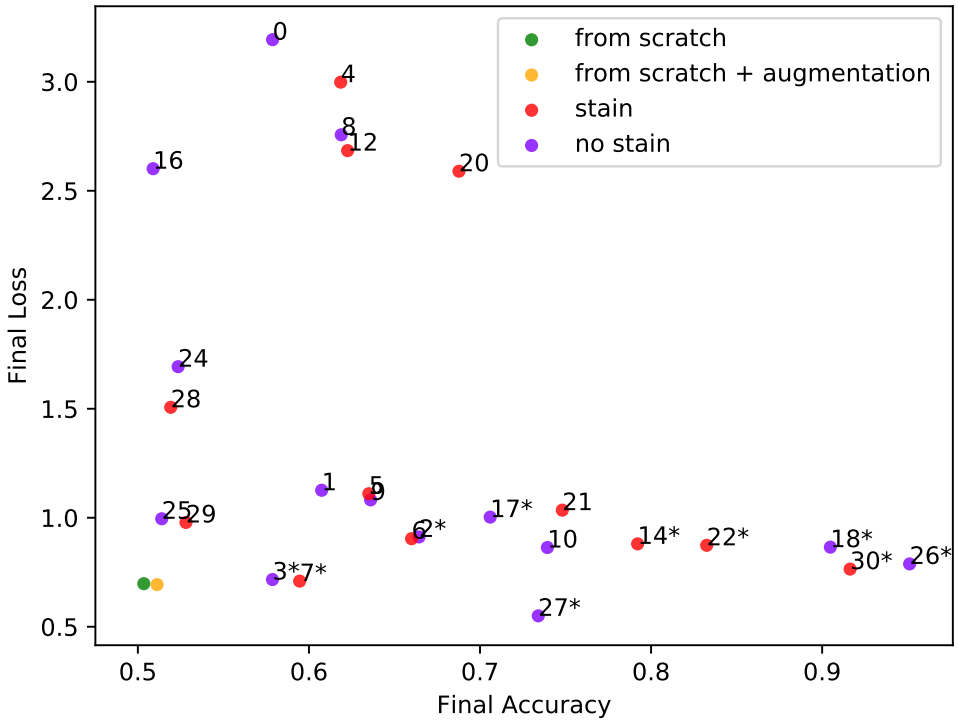


FIGURE 5.5: Stain normalization vs. None (* = Interrupted job).

T-test As a proof of concept, a t-test analysis is conducted. The experiments results show minimal differences in the final performances when stain normalization is applied. Thus, my hypothesis is that the application of stain normalization does not produce statistically relevant changes in terms of final accuracy and loss.

In order to verify this hypothesis, I draw upon a procedure that, to compare two groups of scores (in this case Accuracy without stain normalization and Accuracy with stain normalization etc.), takes into account the differences between them. The two samples of scores were transformed into a single sample of differences, subtracting a set of scores from the other one. Table 5.3 summarized the data.

If the hypothesis that in the second condition no significant effects are produced is true, then the mean $\bar{y_0}$ detected in the control condition does not differ from the mean $\bar{y_1}$ detected in the experimental condition. Therefore I formulated the null hypothesis as following [96]:

Variables	$\bar{y_0}$	$\bar{y_1}$	$\bar{y_1} - \bar{y_0}$
Fin Accuracy	0.67957	0.65799	-0.02158
Fin Accuracy Val	0.67012	0.65569	-0.01443
Fin Loss	1.41946	1.45946	0.04000
Fin Loss Val	0.78479	0.75824	-0.02655

TABLE 5.3: The means of each variable in the two conditions and the mean of the differences between them (which is equivalent to the difference of the means in the two conditions). Condition 0 represents the control condition (no stain normalization) while the condition 1 represents the experimental one (stain normalization).

$$H_0 : \mu_0 = \mu_1 \tag{5.4}$$

where μ_0 and μ_1 represent the means of the entire populations in the two conditions. This expression is equivalent to:

$$H_0 : \mu_0 - \mu_1 = 0 \tag{5.5}$$

where the mean of the entire population is equal in the two different conditions. Thus, the alternative hypothesis can be written as following:

$$H_1 : \mu_0 \neq \mu_1 \tag{5.6}$$

To verify the evidence for or against the null hypothesis with a significance level $\alpha = 0.95$ I used the t-test of significance, in order to identify the t-score and consequently the P-value [97]. The Student's t-distribution was chosen because of the limited dimension of the sample. In particular, the t-score is computed as following:

$$t = \frac{\bar{y} - \mu_0}{se} \tag{5.7}$$

where

$$se = \frac{s}{\sqrt{n}} \tag{5.8}$$

In Table 5.4, the results of the calculations related to the difference between condition 0 and condition 1 are shown.

The t-score was also computed by another method, which however led to the same result [98]:

$$t = \frac{\bar{x} - \bar{y}}{S_{\bar{x}-\bar{y}}} \tag{5.9}$$

where

```
        One Sample t-test

data:  ACCDIFF
t = -1.2367, df = 11, p-value = 0.242
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 -0.05999789  0.01682983
sample estimates:
  mean of x
-0.02158403
```

(A) Training Accuracy.

```
        One Sample t-test

data:  ACCVALDIFF
t = -0.8286, df = 11, p-value = 0.425
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 -0.05277237  0.02390673
sample estimates:
  mean of x
-0.01443282
```

(B) Validation Accuracy.

```
        One Sample t-test

data:  FINLOSSDIFF
t = 1.8537, df = 11, p-value = 0.09077
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 -0.007494459  0.087500190
sample estimates:
 mean of x
0.04000287
```

(C) Training Loss.

```
        One Sample t-test

data:  FINLOSSVALDIFF
t = -1.1411, df = 11, p-value = 0.2781
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 -0.07776888  0.02466383
sample estimates:
  mean of x
-0.02655253
```

(D) Validation Loss.

FIGURE 5.6: Screenshot of R when computing the t-score for the 4 variables.

Differences	*se*	*me*	*r*	*t-score*	*P-value*	*ci* $(1-\beta=0.95)$
Fin Accuracy	0.01745	0.03841	0.913	-1.2367	0.242	(-0.06,0.02)
Fin Accuracy Val	0.01742	0.03834	0.897	-0.8286	0.425	(-0.05,0.02)
Fin Loss	0.02158	0.04750	0.998	1.8537	0.091	(-0.01,0.09)
Fin Loss Val	0.02327	0.05122	0.898	-1.1411	0.278	(-0.08,0.02)

TABLE 5.4: Results of the calculations related to the difference between condition 0 and condition 1. Standard error (*se*), margin of error (*me*), Correlation index (*r*), *t-score*, *P-value* and confidence interval (*ci*) are reported.

$$S^2_{\bar{x}-\bar{y}} = S^2_{\bar{x}} + S^2_{\bar{y}} - 2rS_{\bar{x}}S_{\bar{y}} \tag{5.10}$$

from which

$$S_{\bar{x}-\bar{y}} = \sqrt{S^2_{\bar{x}-\bar{y}}} \tag{5.11}$$

Since all P-values are ≥ 0.05 (i.e. above the α level considering both the tails), it is believed not to reject the null hypothesis at the origin of the present verification. The stain normalization condition does not produce statistically significant effects on the four variables taken into consideration.

The results of the earlier verification have been further checked using the statistical software R as can be seen in Figure 5.6.

In conclusion, stain normalization is unnecessary to the end of my transfer learning approach.

Rescale

The third hyperparameter to choose is the rescale factor. The objects of interests of the GlaS dataset are glands, which are represented with a different scale with respect to the cells of the Camelyon16 dataset. Since the model is pretrained on the Camelyon16 dataset, here I want to evaluate if the different scale factor has some influence on the final performances.
I tried to rescale the GlaS dataset by a $2\times$ factor. If before from each image the extracted 512×512 tile was only one, after the rescaling operation, the tiles were almost 4. Thus, this operation consists in a sort of data augmentation.
The first indirect effect on the experiments was the increase of the data. This led the jobs with both the rescaling and data augmentation preprocessing to crash in the majority of the cases.
In Figure 5.7, the violet points represent the small number of models with a rescaling preprocessing that did not crash. Even if their final loss is lower than the others, their accuracy is really low.
In Figure 5.8, only the models trained with augmented and not stain-normalized data are shown. Here is more evident that most of the experiments which used both data augmentation and rescaling crashed or provided really low accuracy.

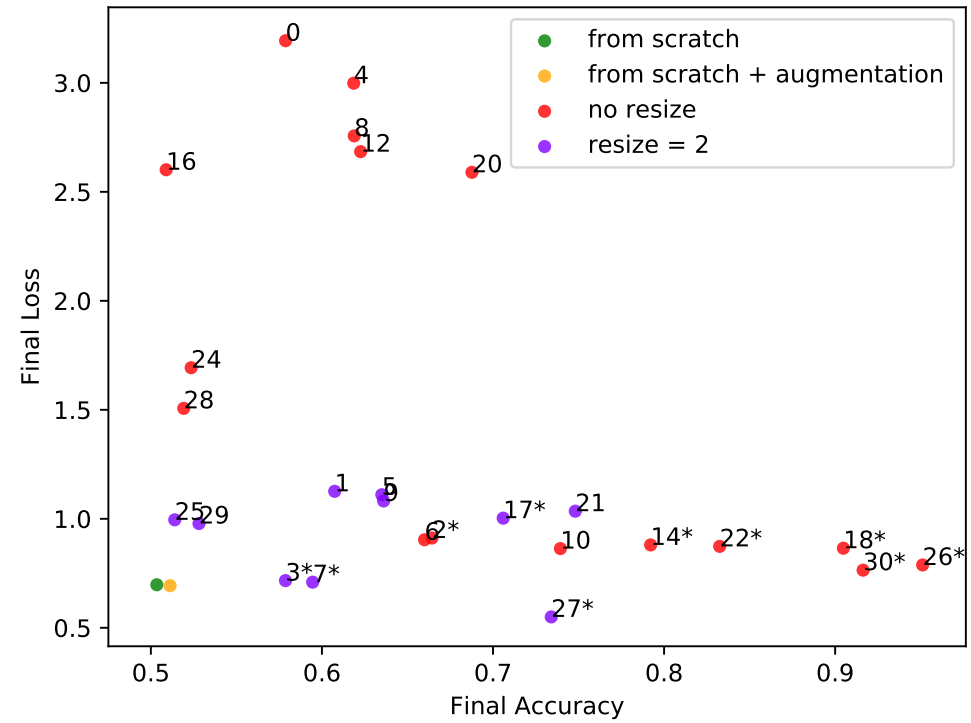


FIGURE 5.7: Resize ×2 vs. None (* = Interrupted job).

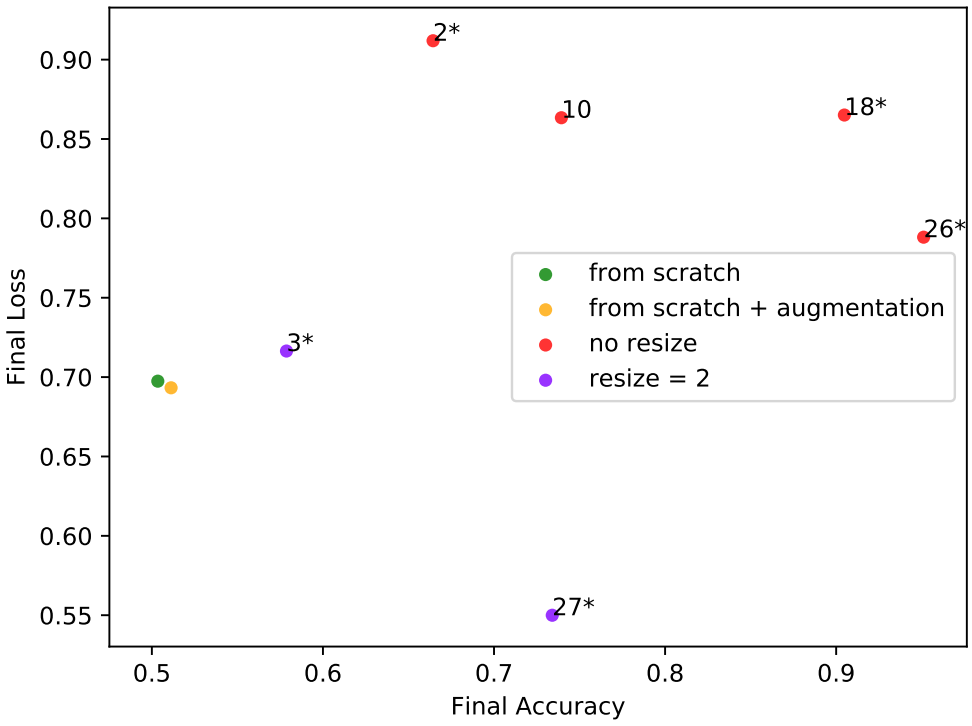


FIGURE 5.8: Resize ×2 vs. None (* = Interrupted job) with data augmentation and no stain normalization.

Number of Frozen Layers

Last but not least, the number of layers to be frozen at the fine-tuning phase must be chosen. In fact, the fine-tuning will start from the weights obtained from the training of UNet on the Camelyon16. Some of these weights can be kept as they are and others can be "fine-tuned". Generally the weights belonging to the the first layers express more general features which can be reused for the new task. Instead, as the depth of the layer increases, the features become even more specific for the original task. Thus the related weights are more likely to be fine-tuned in order to adapt them to the new task.
Taking into account these considerations, an analysis is performed to find the most suitable number of layers to be frozen. In Figure 5.9, the models are highlighted using different colors according to the number of frozen layers used during their fine-tuning.
In Figure 5.10, the models of the previous scatter plot are selected according to the considerations made in the previous paragraphs: only model with data augmentation, no stain normalization and no rescaling are considered. It is evident that there is a trend in the performances: they gradually increase when the number of frozen layers decrease. This is coherent with the initial hypothesis: if the number of frozen layers is too high, the performances decrease probably because deeper features are too specific to the source task.
In conclusion, the best choice is to avoid the freeze of the layers.

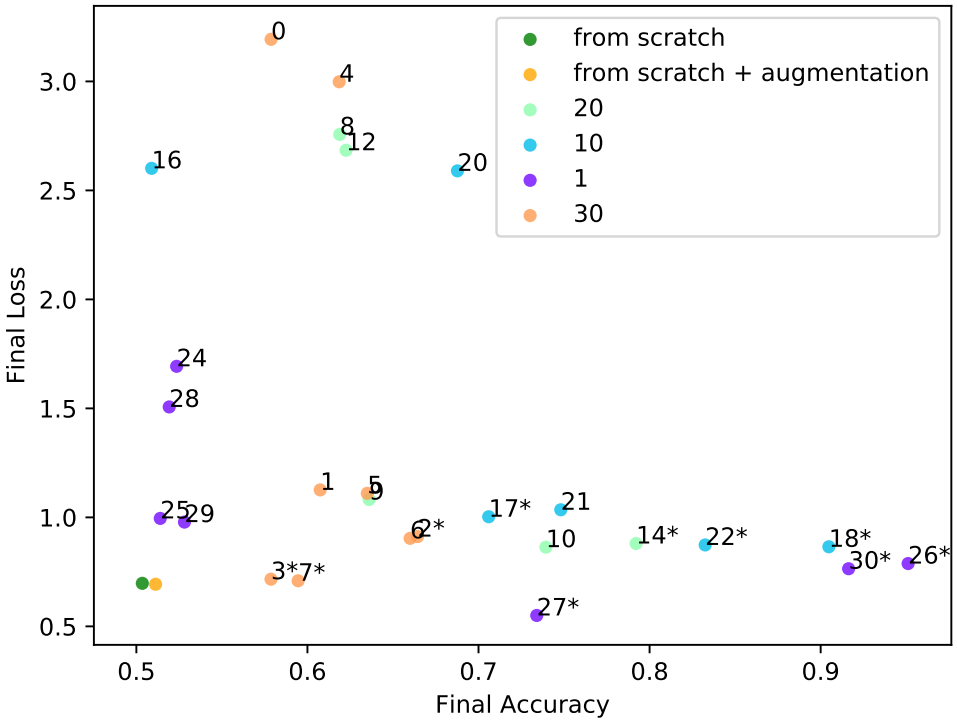


FIGURE 5.9: Different number of frozen layers (* = Interrupted job).

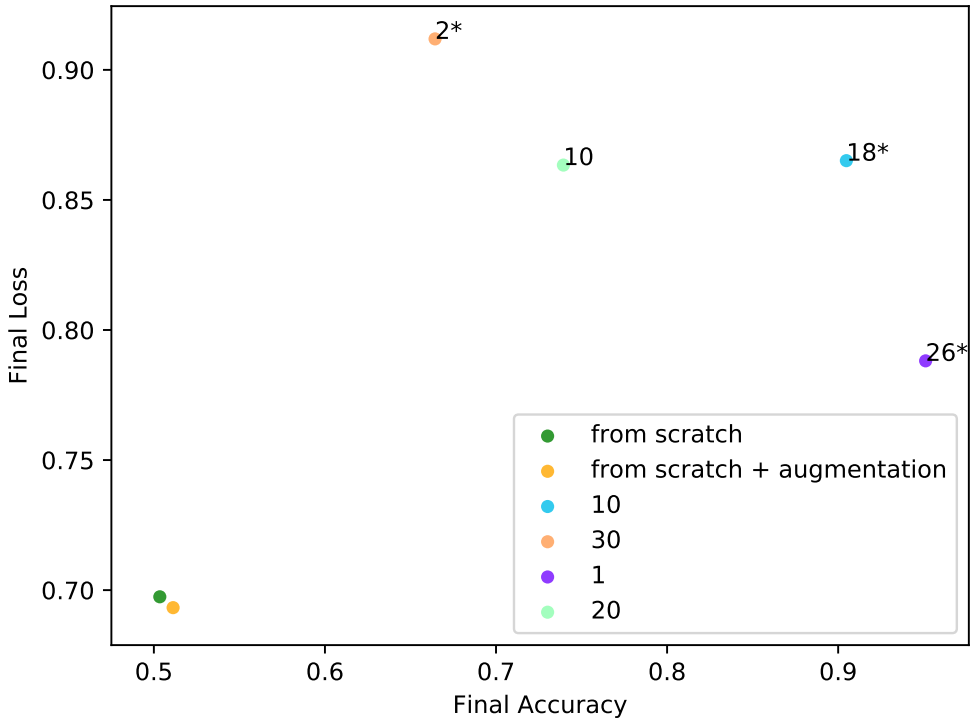


FIGURE 5.10: Different number of frozen layers (* = Interrupted job) with data augmentation, no stain normalization and no rescaling.

Computational Analysis

In Figure 5.11 and 5.12 the accuracy and the loss for all the trained models as function of the training epoch are show. This plots highlight the behavior of the different models for every epochs, while the considerations of the previous paragraph were drawn only for the final value of accuracy and loss reached.
From both the Figures we can see that each model reaches different final performances in a really different manner. There are models able to converge in few epochs but also models that require more cycles to reach the convergence.
As already stated, the computational resources can be an important issues. Thus it is preferable that a model is able to reach good performances is less epochs. However, a note of caution: a small number of epochs does not mean that the correspondent time is less then the one required for training one other model for a larger number of epochs. In fact, as seen in Section 4.5.5, an epoch represents an iteration over the entire training data and label sets. Thus, if the number of training sample is higher, also the time to complete an epoch will be higher. Also the batch size can influence the time required to complete an epoch. In fact, it represents the number of training samples taken into account before each gradient update. The more times the gradient is updated, the more time will be required for a single epoch.

Thus it is necessary to analyze each model in terms of the computational time required for its convergence. For this purpose, an additional callback is added to the *fit* function of *Keras* in order to save the time required to complete each epoch. At the end, the cumulative time is computed for each epoch obtaining the graphic in Figure 5.13.

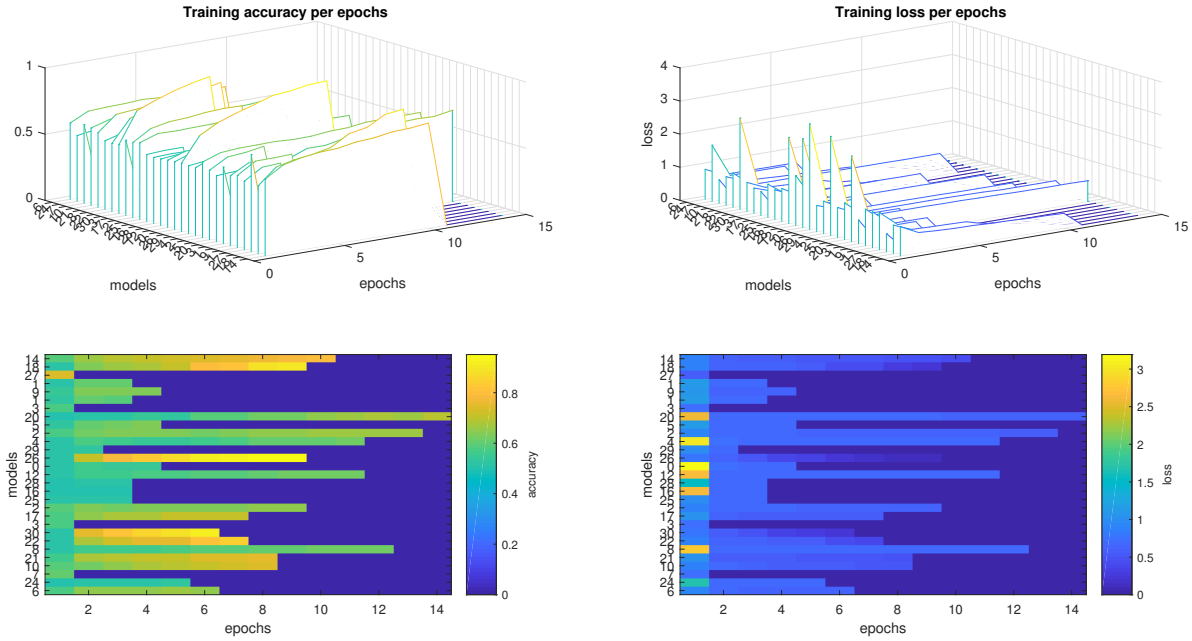


FIGURE 5.11: Training performances of all the models. In the left column the trend of the accuracies over epochs is reported while in the right one the trend of the loss.

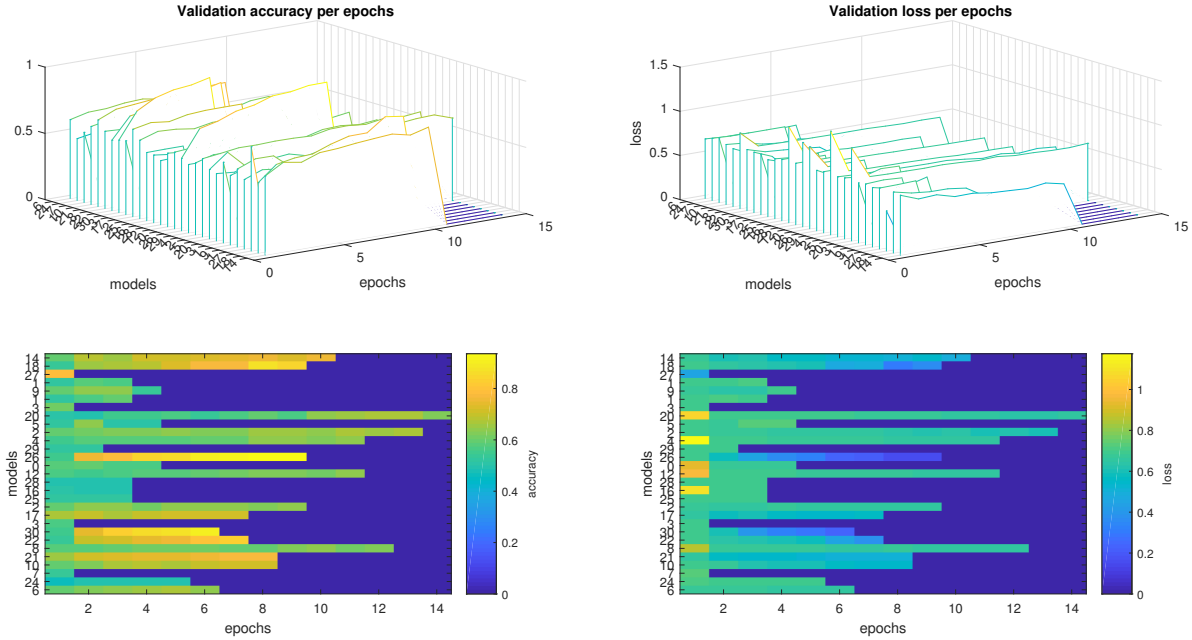


FIGURE 5.12: Validation performances of all the models. In the left column the trend of the accuracies over epochs is reported while in the right one the trend of the loss.

Some jobs have been killed because of *walltime* policy on the Cluster. The cluster scheduler allows only 10 days of computing in terms of absolute time (not effectively computing time), after that deadline the jobs are killed. Therefore, most of the models did not complete the training. The peaks in the graphics (yellow entries in the matrix) indicates the total cpu-time effectively spent by the algorithm which is not bigger than walltime declared per job. It can be noticed that those peaks have size around 6×10^5 seconds, i.e. around 7 days. These jobs are probably been killed by the scheduler. Some other jobs may be killed because of some cluster policy, such as walltime, resources and priorities.

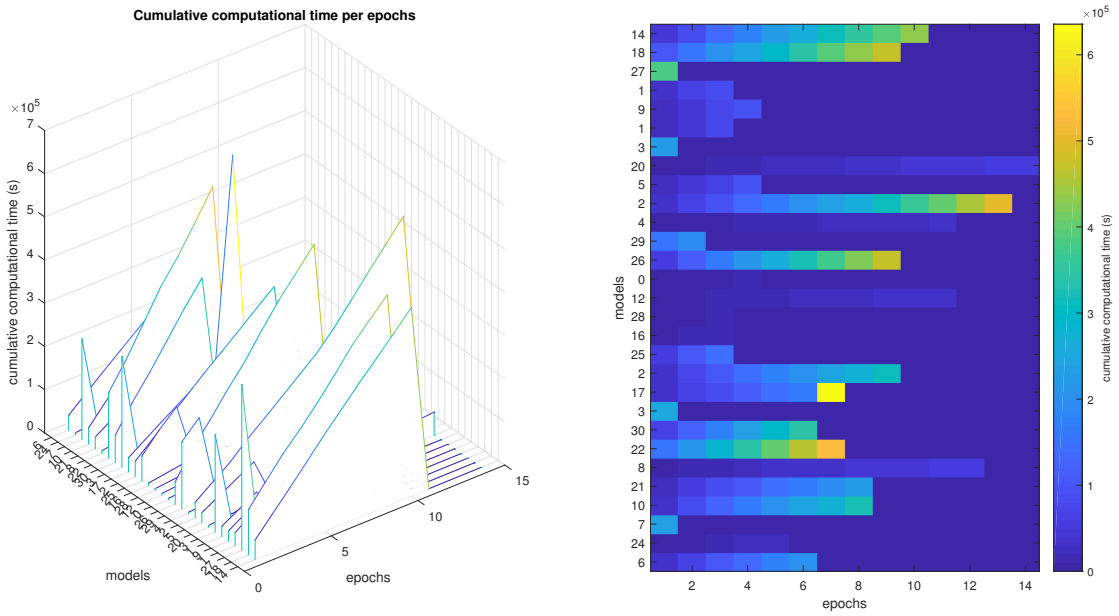


FIGURE 5.13: Cumulative time per epoch of each model.

Regarding the memory load, the maximal virtual memory used was 25 GB, but most of the jobs used less than 20 GB. UNet is a really small CNN with respect to other models in literature, leading to big advantages in terms of memory.

Usually, the memory required for training a CNN is used for storing the following data [8]:

- model parameters (weights);
- feature maps (intermediate matrices generated in the forward phase);
- gradient maps (intermediate matrices generated in the backward phase);
- workspace (temporary data).

Authors of [99] showed that features maps are the most memory-expensive data and that the fraction of memory needed for allocating them grows as the number of layers increase. It happens because the gradient update is layer-wise, i.e. during the backpropagation phase features maps are taken into account [99]. This means that features maps should be tracked and kept in memory until backpropagation

[8] https://medium.com/syncedreview/how-to-train-a-very-large-and-deep-model-on-one-gpu-7b7edfe2d072

ends. The fraction of memory required for features maps is more relevant in the features extraction layers and decrease in the classification ones [99]. The opposite for weights which are more relevant in the classification layers due to the fully connectivity [99]. Their computational load is in any case negligible with respect to the features maps and workspace ones [99].

In conclusion, a CNN with a small number of layers like UNet does not require big computational resources. Moreover, the number of parameters to train is not a decisive factor in terms of memory. Thus, in our case, freezing more layers would not improve the performances in that sense.

Comparison with UNet trained on GlaS from scratch

The performances are extremely promising if compared with the ones provided by UNet trained from scratch both with and without data augmentation. In Table 5.5 a comparison of the performances of the UNet trained with a non-random initialization of the weights and two reference models is presented. In particular, the transfer learning model is compared with UNet trained starting from a random initialization of the weights both with and without data augmentation.

model	acc	loss	val acc	val loss
From scratch	0.50347	0.69744	0.49040	0.69310
From scratch + data aug	0.51124	**0.69326**	0.53064	0.69307
Best model	**0.95113**	0.78814	**0.93520**	**0.68563**

TABLE 5.5: Comparison of the performances between the best new model and UNet trained from scratch with and without data augmentation.

In terms of accuracy, all the new models trained with transfer learning and data augmentation outperformed the same model trained from scratch both with and without data augmentation. This was particularly evident in the scatter plots of the previous section, where the two reference models (orange and green points) where located in the bottom-left corner of the 2D plane.

In terms of loss measures this is less true, but is well worth thinking carefully about it. The two models trained from scratch present a really low loss but at the same time an accuracy close to the one of the random classifier. A so-evident discrepancy between the loss and the accuracy performances can occur in many situations due to the deeply different meaning they have.
A probable interpretation is that these two models are really confident when they classify correctly the samples, while at the same time they are really low confident when they classify wrongly the samples. As explained in Section 5.1, the logarithmic loss function humbly penalizes an uncertain wrong prediction while hugely penalizes a certain wrong one. In the same way, it rewards hugely a certain correct prediction and rewards humbly an uncertain one. According to this interpretation, these two models provide certain predictions when these are actually correct, and uncertain predictions when these are actually wrong.
Moreover I chose as cost function the loss and not the accuracy. Thus, during training, the model tries to minimize the loss and not to maximize the accuracy. Minimizing the loss means not only correct predictions, but most of all getting even more confident in the correct predictions and less confident in the wrong ones. In this situation accuracy can even not improve while the loss is decreasing leading not to a better model but to a more robust one.

By the way the transfer learning results are even more promising if considering that most of the best models did not complete the training in the 10 days allowed by the cluster scheduler. Thus they have still some margins of improvements.

The best model chosen for the comparison is the one trained with augmented, not stain normalized, not rescaled data without freezing any layer. His hyperparameters are summarized in Table 5.6.

Augmentation	Stain Normalization	Resize	Nr of frozen layers
True	False	False	0

TABLE 5.6: Hyperparameters of the best new model (model 26 in the scatter plots).

Test on an independent set of data

On the best model, an additional test is done on a completely independent test set of GlaS data. These data are even not augmented. The main objective is to evaluate the model on never-seen images, exclude the risk of overfitting, and validate the results provided by the *fit* function of *Keras* which automatically splits the training set into two additional training and validation subsets.

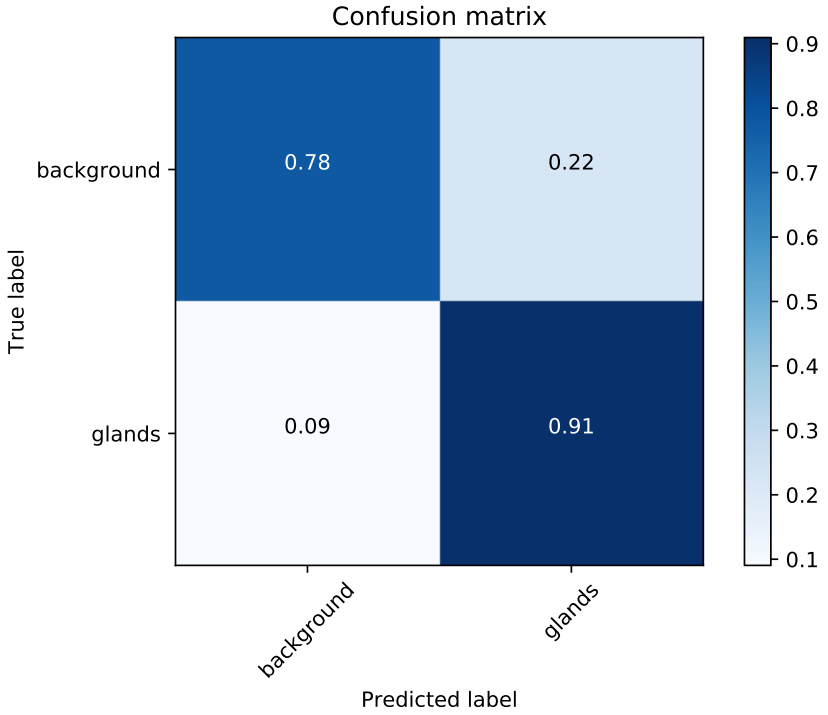


FIGURE 5.14: Confusion matrix of the best new model computed on a completely independent test set.

The resulting confusion matrix is reported in Figure 5.14 while the correspondent results are provided in Table 5.7. The confusion matrix and the consequent performances are extracted from all the images simultaneously and do not represent a mean of the confusion matrix computed for each image.

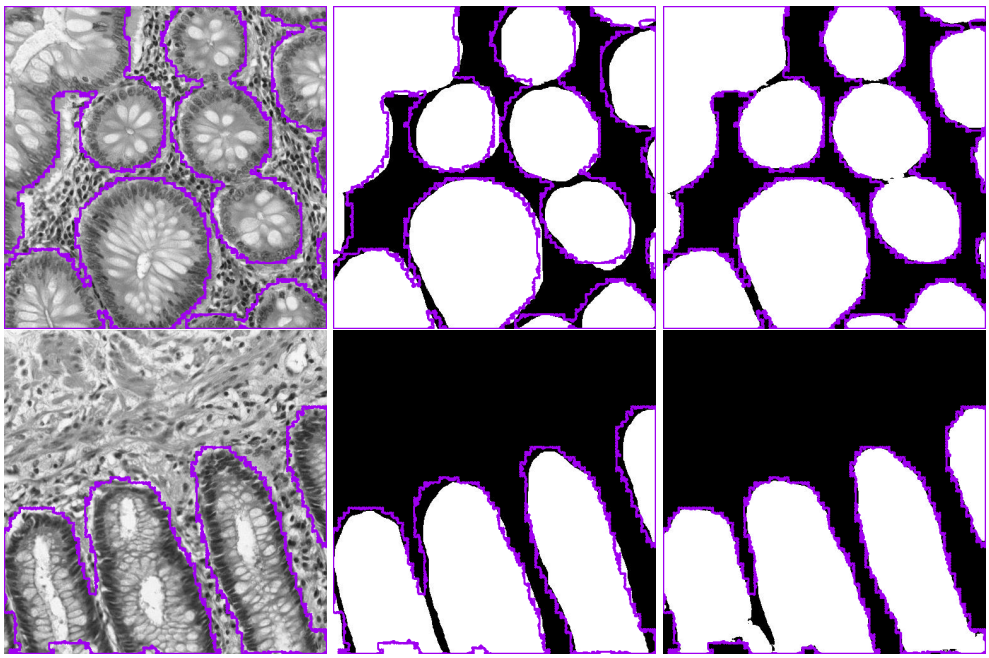
From the data it can be deduced that UNet classifies better glands than the background. UNet tends to predict glands when the true label is the background and not

the opposite. This can be seen also in the Figures 5.15, where it is evident that UNet oversegments glands (the violet edges are bigger than the ground truth).
In fact, the recall value, which express the proportion of actual positives that are correctly classified, is really good while precision, i.e. the proportion of positive predictions that are actually correct, is much more lower because there are lots of false positive.
These results are not as good as the ones computed by the *fit* function but are still really promising. Moreover, if we consider that the majority of the misclassified pixels are false positive (which, in a biomedical setting is the best situation: is better to classify the background as glands instead of missing a gland) and these FP represent only an oversegmentation of glands and are not located in other area of the image.

Accuracy	Misclassification Error	Precision	Recall	F1 score
84.5 %	15.5 %	80.53 %	91 %	85.4 %

TABLE 5.7: Performances over all pixels of the best new model computed on a completely independent test set.

In Figure 5.15, some examples of prediction of the best new model are shown. In the left column there are the original grayscale images, in the central one the ground truth masks and in the left one the predictions. Every image presents the edges of the predicted mask superimposed in order to qualitatively evaluate the performances. Looking carefully the ground truth labels, we can deduce that the model is able to classify very well glands while the main difficulties are in the glands boarders. In fact, usually glands present a tubular structure in which it can be distinguished the internal lumen surrounded by epithelial cells as described in Section 3.4. This particular arrangement of the epithelial nuclei around the lumen can be used to identify clearly the lumen but the boarder itself results more difficult to delineate.



(A) Original image (B) Ground truth mask (C) Predicted mask

FIGURE 5.15: Examples of prediction of the best new model. The violet edges are extracted from the mask predicted by the new model.

5.3.2 ISBI12 dataset

A grid search is performed in order to find the best set of hyperparameters for the ISBI12 dataset. For each combinations of hyperparameters a specific job is generated where a UNet model is fine-tuned with a given number of frozen layers and certain preprocessing steps.
Exploiting the experience had with the GlaS dataset, the hyperparameter to fix is only the number of layers to freeze. Data augmentation is performed due to his decisive role played with the GlaS dataset. No rescale factor is applied due to the really bad performances and memory issues obtained with the GlaS Moreover, stain normalization is useless because the images are acquired without any stain.
The hyperparameters tested in this analysis are summarized in Table 5.8.

Parameter	Values
data augmentation	True
stain normalization	False
resize	$\times 1$
number of frozen layers	0, 10, 20, 30

TABLE 5.8: Hyperparameters for the Fine Tuning of UNet with ISBI12 data.

All the 4 tested models converge to a training accuracy of 85.37% and a validation accuracy of 85.75% which corresponds to a completely blank prediction. This is possible because here the new task is much more different than the previous one: the task is more a boundary segmentation instead of a semantic segmentation as can be seen in Figure 5.16 (image taken from [9]).
In literature this task, generally is solved with an ad-hoc pipeline that allows for the segmentation of particular ROIs such as boarders.

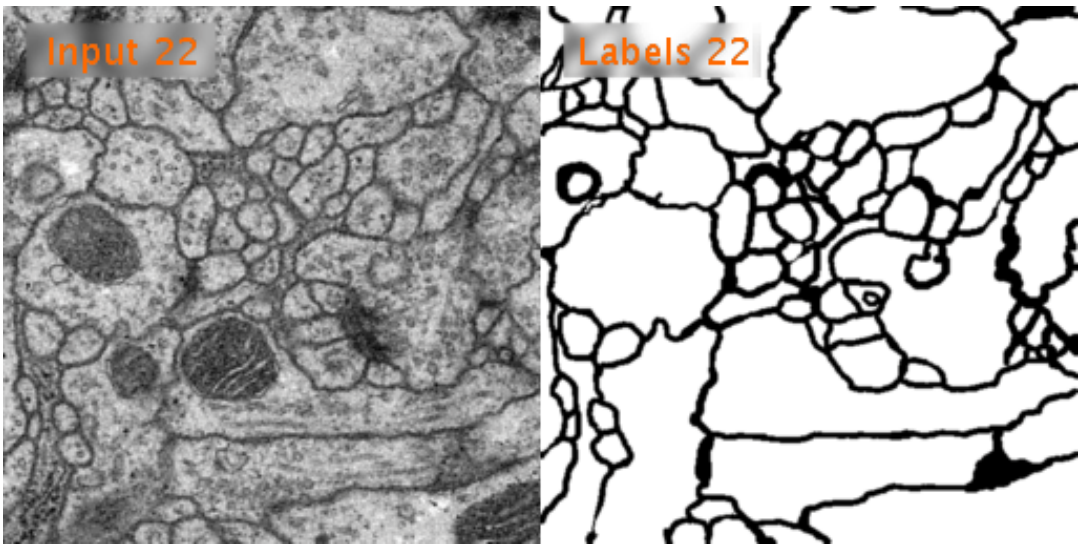
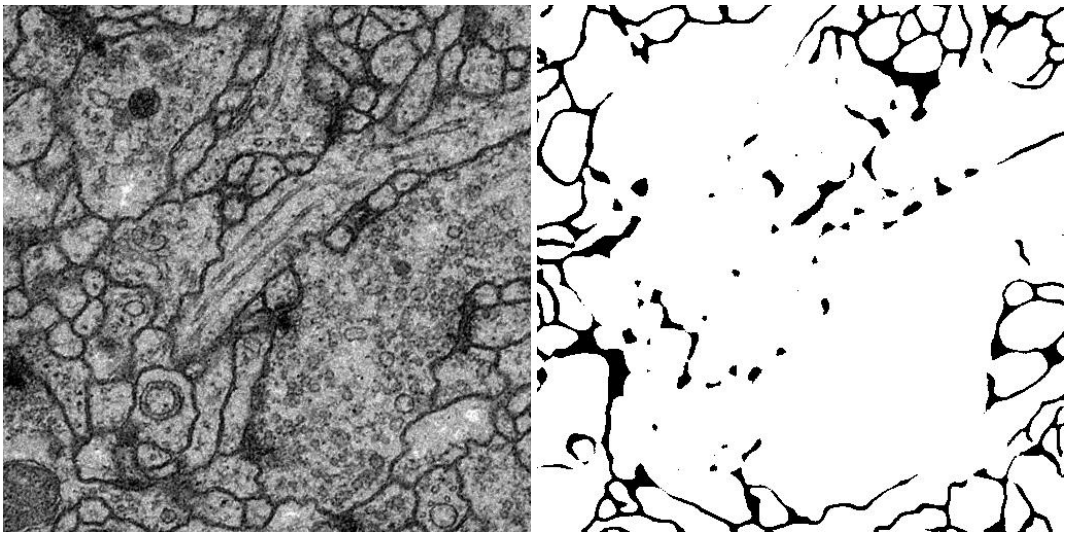


FIGURE 5.16: Example of image of the ISBI12 dataset and the corresponding mask taken from the challenge website.

[9] http://brainiac2.mit.edu/isbi_challenge/home

Comparison with UNet trained on ISBI12 from scratch

On the other hand, training UNet from scratch provides better results with a training accuracy of 81.42% that is improved to 89.01% by data augmentation. In this case the model provides reasonable masks as outputs which led to hope in better even results if the number of training images will increase. An example of such output is provided in Figure 5.17.
Probably the Camelyon16 initialization of the weights led the model to converge in a local minimum which corresponds to totally blank predictions. This minimum is avoided if the training starts from a random initialization of the weights.



(A) Original image. (B) Predicted mask.

FIGURE 5.17: Example of output of UNet trained from scartch with augmented ISBI12 data.

Chapter 6

Conclusions and Future Work

In this chapter, the most important results are commented, the limits of this work are presented and some indications are stated for future improvement. Finally the contributions and novelties of this thesis are stated.

6.1 Discussion of the Results

Within this thesis, the benefits and drawbacks of transfer learning and data augmentation for image semantic segmentation have been analyzed.
The major benefits were detected in the GlaS application, where the combination of data augmentation and transfer learning allowed to outperform the same model trained from scratch both with and without data augmentation. Really good performances were obtained in a reasonable computational time using an exiguous portion of memory. This is a really important result because huge computational resources are not within everyone's means, especially when dealing with big biomedical datasets. Moreover, most of the best models were not able to finish their training due to the *walltime* policy of the cluster, which led to hope for good margin of improvements. One other secondary result was the fact that stain normalization does not affect significantly the performances as stated in literature. This is true for my transfer learning pipeline but it would be a big benefit also in other applications.

When it comes to the ISBI12, the results change dramatically. At first sight, the performances are quite good in a quantitative sense, but going deep with the qualitative analysis of the test outputs I discovered that the model fell into a local minimum. In fact, all the predictions were blank images. Instead the model trained from scratch with an augmented dataset provides better results. Probably the non-random initialization of the weights led the model to extract features which were to much connected to the segmentation of ROIs and not of thin objects like membrane boarders.

In conclusion, I can say that transfer learning is an optimal choice for improving the performances of a model when the computational resources and big datasets are insufficient or not available. This approach is particularly successful when the novel task is similar to the original one and can be an obstacle when the novel task is too much different.

Anyhow, to my knowledge, such approach represents a novelty not only in the biomedical field as well as in many other fields of computer vision.

6.2 Limitations

The biggest limitations I came upon while working on this thesis was the absence of big, fully-annotated and freely available histopathological datasets. The three

datasets I used were the only ones that matched these requirements and I could find. Often researcher are constrained to build themselves the ground truth. This operation is not only tedious and time-consuming but also susceptible to mistakes which will led the model to learn wrongly from the data.

The second major limitation I had was related to the computational resources. The HPC cluster was really useful but the 10 days of walltime policy was too much constrain. Similarly, one other big constrain was the presence of only two GPU nodes which generates long queues for their use.

These two limitations are the ones to cut down if academic research wants to challenge big industries like Google, Amazon or Microsoft. Today if an industry owns huge a computational power and has access to the user's information (huge datasets!!) is able to build whatever artificial intelligence it wants. I believe that transfer learning and data augmentation will reduce the need of such factors to provide good scientific results.

6.3 Future Directions

Probably, some of the future directions will be:

- test new biomedical and not-biomedical datasets;
- test novel tasks with an increasing level of difference from the source task;
- do more testing cases for example on the parameters of the training phase, the cost function and so on;
- evaluate the influence of the number of augmented samples and the type of data augmentation;
- resume the best models and finish their training;
- Find a way to transfer the knowledge also to the boundary segmentation task.

6.4 Contributions and Novelty of this work

Theoretical contribution: **Transfer Learning** applied to **Semantic Segmentation** represents, to my knowledge, an **absolute novelty** not only in the biomedical field, but also in general in the computer vision one.

Practical contribution: Provide a **Python code** for an easy testing of Fine Tuning and Data Augmentation on pre-trained `Keras` models. The code will be available on my GitHub page as soon as possible.

Experimental contribution: Evaluation of the benefits of Transfer Learning and Data Augmentation on **real-world** histopathological datasets.

Bibliography

[1] Seunghoon Hong et al. "Learning transferrable knowledge for semantic segmentation with deep convolutional neural network". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 3204–3212.

[2] Judy Hoffman et al. "LSDA: Large scale detection through adaptation". In: *Advances in Neural Information Processing Systems*. 2014, pp. 3536–3544.

[3] Standford edu. *Convolutional Neural Networks (CNNs / ConvNets)*. http://cs231n.github.io/convolutional-networks/#conv.

[4] Qiang Yang. *Introduction to Deep Learning*. https://slideplayer.com/slide/11909715/.

[5] David H Hubel and Torsten N Wiesel. "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex". In: *The Journal of physiology* 160.1 (1962), pp. 106–154.

[6] Alberto Garcia-Garcia et al. "A Review on Deep Learning Techniques Applied to Semantic Segmentation". In: *arXiv preprint arXiv:1704.06857* (2017).

[7] Yun Liu et al. "Detecting cancer metastases on gigapixel pathology images". In: *arXiv preprint arXiv:1703.02442* (2017).

[8] Meet Pragnesh Shah. *Semantic Segmentation using Fully Convolutional Networks over the years*. https://meetshah1995.github.io/semantic-segmentation/deep-learning/pytorch/visdom/2017/06/01/semantic-segmentation-over-the-years.html. 2017.

[9] Sasank Chilamkurthy. *A 2017 Guide to Semantic Segmentation with Deep Learning*. http://blog.qure.ai/notes/semantic-segmentation-deep-learning-review. 2017.

[10] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2015, pp. 234–241.

[11] Dan Ciresan et al. "Deep neural networks segment neuronal membranes in electron microscopy images". In: *Advances in neural information processing systems*. 2012, pp. 2843–2851.

[12] Yan Xu et al. "Large scale tissue histopathology image classification, segmentation, and visualization via deep convolutional activation features". In: *BMC bioinformatics* 18.1 (2017), p. 281.

[13] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3431–3440.

[14] Adam Paszke et al. "Enet: A deep neural network architecture for real-time semantic segmentation". In: *arXiv preprint arXiv:1606.02147* (2016).

[15] Abhishek Chaurasia and Eugenio Culurciello. "LinkNet: Exploiting Encoder Representations for Efficient Semantic Segmentation". In: *arXiv preprint arXiv:1707.03718* (2017).

[16] Md Amirul Islam et al. "Gated feedback refinement network for dense image labeling". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2017, pp. 4877–4885.

[17] Guosheng Lin et al. "Refinenet: Multi-path refinement networks with identity mappings for high-resolution semantic segmentation". In: *arXiv preprint arXiv:1611.06612* (2016).

[18] Chao Peng et al. "Large Kernel Matters–Improve Semantic Segmentation by Global Convolutional Network". In: *arXiv preprint arXiv:1703.02719* (2017).

[19] Simon Jégou et al. "The one hundred layers tiramisu: Fully convolutional DenseNets for semantic segmentation". In: *arXiv preprint arXiv:1611.09326* (2016).

[20] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. "Segnet: A deep convolutional encoder-decoder architecture for image segmentation". In: *arXiv preprint arXiv:1511.00561* (2015).

[21] Alexander Jung. "imgaug, a library for image augmentation in machine learning experiments". In: 2018.

[22] Sinno Jialin Pan and Qiang Yang. "A survey on transfer learning". In: *IEEE Transactions on knowledge and data engineering* 22.10 (2010), pp. 1345–1359.

[23] Jason Yosinski et al. "How transferable are features in deep neural networks?" In: *Advances in neural information processing systems*. 2014, pp. 3320–3328.

[24] Edward Kim and Zubair Baloch. "A deep semantic mobile application for thyroid cytopathology". In:

[25] *Camelyon16-*. 2016. URL: https://camelyon16.grand-challenge.org/results/.

[26] Geert J. S. Litjens et al. "A Survey on Deep Learning in Medical Image Analysis". In: *CoRR* abs/1702.05747 (2017). URL: http://arxiv.org/abs/1702.05747.

[27] Dan Claudio Ciresan et al. "Mitosis detection in breast cancer histology images with deep neural networks". In: (2013).

[28] Angel Alfonso Cruz-Roa et al. "A deep learning architecture for image representation, visual interpretability and automated basal-cell carcinoma cancer detection". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, Berlin, Heidelberg. 2013, pp. 403–410.

[29] Christopher D Malon and Eric Cosatto. "Classification of mitotic figures with convolutional neural networks and seeded blob features". In: *Journal of pathology informatics* 4 (2013).

[30] Haibo Wang et al. "Mitosis detection in breast cancer pathology images by combining handcrafted and convolutional neural network features". In: *Journal of Medical Imaging* 1.3 (2014), pp. 034003–034003.

[31] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *CoRR* abs/1505.04597 (2015). URL: http://arxiv.org/abs/1505.04597.

[32] Anat Shkolyar et al. "Automatic detection of cell divisions (mitosis) in live-imaging microscopy images using convolutional neural networks". In: *Engineering in Medicine and Biology Society (EMBC), 2015 37th Annual International Conference of the IEEE*. IEEE. 2015, pp. 743–746.

[33] Youyi Song et al. "Accurate segmentation of cervical cytoplasm and nuclei based on multiscale convolutional network and graph partitioning". In: *IEEE Transactions on Biomedical Engineering* 62.10 (2015), pp. 2421–2433.

[34] Yuanpu Xie. "Xiangfei Kong, Fuyong Xing, Fujun Liu, Hai Su, and Lin Yang,"Deep voting: A robust approach toward nucleus localization in microscopy images,"" in: MICCAI. 2015.

[35] Yuanpu Xie et al. "Beyond classification: structured regression for robust cell detection using convolutional neural network". In: *Medical image computing and computer-assisted intervention: MICCAI... International Conference on Medical Image Computing and Computer-Assisted Intervention*. Vol. 9351. NIH Public Access. 2015, p. 358.

[36] Saad Ullah Akram et al. "Cell proposal network for microscopy image analysis". In: *Image Processing (ICIP), 2016 IEEE International Conference on*. IEEE. 2016, pp. 3199–3203.

[37] Shadi Albarqouni et al. "Aggnet: deep learning from crowds for mitosis detection in breast cancer histology images". In: *IEEE transactions on medical imaging* 35.5 (2016), pp. 1313–1321.

[38] Stefan Bauer et al. "Multi-organ cancer classification and survival analysis". In: *arXiv preprint arXiv:1606.00897* (2016).

[39] Zhimin Gao et al. "Hep-2 cell image classification with deep convolutional neural networks". In: *IEEE journal of biomedical and health informatics* 21.2 (2017), pp. 416–428.

[40] Xian-Hua Han, Jianmei Lei, and Yen-Wei Chen. "HEp-2 cell classification using K-support spatial pooling in deep CNNs". In: *International Workshop on Large-Scale Annotation of Biomedical Data and Expert Label Synthesis*. Springer. 2016, pp. 3–11.

[41] Muhammad Nasim Kashif et al. "Handcrafted features with convolutional neural networks for detection of tumor cells in histology images". In: *Biomedical Imaging (ISBI), 2016 IEEE 13th International Symposium on*. IEEE. 2016, pp. 1029–1032.

[42] Yunxiang Mao and Zhaozheng Yin. "A hierarchical convolutional neural network for mitosis detection in phase-contrast microscopy images". In: *MICCAI 2016: Medical Image Computing and Computer-Assisted Intervention–MICCAI 2016* (2016).

[43] Manish Mishra et al. "Structure-based assessment of cancerous mitochondria using deep networks". In: *Biomedical Imaging (ISBI), 2016 IEEE 13th International Symposium on*. IEEE. 2016, pp. 545–548.

[44] Ha Tran Hong Phan et al. "Transfer learning of a convolutional neural network for HEp-2 cell image classification". In: *Biomedical Imaging (ISBI), 2016 IEEE 13th International Symposium on*. IEEE. 2016, pp. 1208–1211.

[45] David Romo-Bucheli et al. "Automated tubule nuclei quantification and correlation with oncotype DX risk categories in ER+ breast cancer whole slide images". In: *Scientific reports* 6 (2016).

[46] Korsuk Sirinukunwattana et al. "Locality sensitive deep learning for detection and classification of nuclei in routine colon cancer histology images". In: *IEEE transactions on medical imaging* 35.5 (2016), pp. 1196–1206.

[47] Riku Turkki et al. "Antibody-supervised deep learning for quantification of tumor-infiltrating immune cells in hematoxylin and eosin stained breast cancer samples". In: *Journal of pathology informatics* 7 (2016).

[48] Sheng Wang et al. "Subtype cell detection with an accelerated deep convolution neural network". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2016, pp. 640–648.

[49] Fuyong Xing, Yuanpu Xie, and Lin Yang. "An automatic learning-based framework for robust nucleus segmentation". In: *IEEE transactions on medical imaging* 35.2 (2016), pp. 550–566.

[50] Jun Xu et al. "Stacked sparse autoencoder (SSAE) for nuclei detection on breast cancer histopathology images". In: *IEEE transactions on medical imaging* 35.1 (2016), pp. 119–130.

[51] Zheng Xu and Junzhou Huang. "Detecting 10,000 Cells in one second". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2016, pp. 676–684.

[52] Lin Yang et al. "3d segmentation of glial cells using fully convolutional networks and k-terminal cut". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer International Publishing. 2016, pp. 658–666.

[53] Jianwei Zhao et al. "Automatic detection and classification of leukocytes using convolutional neural networks". In: *Medical & biological engineering & computing* (2016), pp. 1–15.

[54] Jiawen Yao et al. "Imaging biomarker discovery for lung cancer survival prediction". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2016, pp. 649–657.

[55] Youyi Song et al. "Accurate cervical cell segmentation from overlapping clumps in pap smear images". In: *IEEE transactions on medical imaging* 36.1 (2017), pp. 288–300.

[56] Philipp Kainz, Michael Pfeiffer, and Martin Urschler. "Semantic segmentation of colon glands with deep convolutional neural networks and total variation segmentation". In: *arXiv preprint arXiv:1511.06919* (2015).

[57] Grégory Apou et al. "Detection of lobular structures in normal breast tissue". In: *Computers in biology and medicine* 74 (2016), pp. 91–102.

[58] Aïcha BenTaieb and Ghassan Hamarneh. "Topology Aware Fully Convolutional Networks for Histology Gland Segmentation." In: *MICCAI (2)*. 2016, pp. 460–468.

[59] Aïcha BenTaieb, Jeremy Kawahara, and Ghassan Hamarneh. "Multi-loss convolutional networks for gland analysis in microscopy". In: *Biomedical Imaging (ISBI), 2016 IEEE 13th International Symposium on*. IEEE. 2016, pp. 642–645.

[60] Jianxu Chen et al. "Combining fully convolutional and recurrent neural networks for 3d biomedical image segmentation". In: *Advances in Neural Information Processing Systems*. 2016, pp. 3036–3044.

[61] Michal Drozdzal et al. "The importance of skip connections in biomedical image segmentation". In: *International Workshop on Large-Scale Annotation of Biomedical Data and Expert Label Synthesis*. Springer. 2016, pp. 179–187.

[62] Wenqi Li et al. "Gland segmentation in colon histology images using hand-crafted features and convolutional neural networks". In: *Biomedical Imaging (ISBI), 2016 IEEE 13th International Symposium on*. IEEE. 2016, pp. 1405–1408.

[63] Petteri Teikari et al. "Deep learning convolutional networks for multiphoton microscopy vasculature segmentation". In: *arXiv preprint arXiv:1606.02382* (2016).

[64] Yan Xu et al. "Gland instance segmentation by deep multichannel side supervision". In: *arXiv preprint arXiv:1607.03222* (2016).

[65] Jun Xu et al. "A deep convolutional neural network for segmenting and classifying epithelial and stromal regions in histopathological images". In: *Neurocomputing* 191 (2016), pp. 214–223.

[66] Hao Chen et al. "Dcan: Deep contour-aware networks for accurate gland segmentation". In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2016, pp. 2487–2496.

[67] Angel Cruz-Roa et al. "Automatic detection of invasive ductal carcinoma in whole slide images with convolutional neural networks". In: *SPIE medical imaging*. Vol. 9041. International Society for Optics and Photonics. 2014, pp. 904103–904103.

[68] Yan Xu et al. "Deep learning of feature representation with multiple instance learning for medical image analysis". In: *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE. 2014, pp. 1626–1630.

[69] Mehmet Günhan Ertosun and Daniel L Rubin. "Automated grading of gliomas using deep learning in digital pathology images: A modular approach with ensemble of convolutional neural networks". In: *AMIA Annual Symposium Proceedings*. Vol. 2015. American Medical Informatics Association. 2015, p. 1899.

[70] Dmitrii Bychkov et al. "Deep learning for tissue microarray image-based outcome prediction in patients with colorectal cancer". In: *Medical Imaging* 9791 (2016), p. 979115.

[71] John A Quinn et al. "Deep convolutional neural networks for microscopy-based point of care diagnostics". In: *Machine Learning for Healthcare Conference*. 2016, pp. 271–281.

[72] Hadi Rezaeilouyeh, Ali Mollahosseini, and Mohammad H Mahoor. "Microscopic medical image classification framework via deep learning and shearlet transform". In: *Journal of Medical Imaging* 3.4 (2016), pp. 044501–044501.

[73] Dayong Wang et al. "Deep learning for identifying metastatic breast cancer". In: *arXiv preprint arXiv:1606.05718* (2016).

[74] Hang Chang et al. "Unsupervised Transfer Learning via Multi-Scale Convolutional Sparse Coding for Biomedical Applications". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2017).

[75] Christian Szegedy et al. "Going deeper with convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

[76] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[77] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[78] Babak Ehteshami Bejnordi et al. "Stain specific standardization of whole-slide histopathological images". In: *IEEE transactions on medical imaging* 35.2 (2016), pp. 404–415.

[79] Kyunghyun Paeng et al. "A unified framework for tumor proliferation score prediction in breast histopathology". In: *arXiv preprint arXiv:1612.07180* (2016).

[80] Korsuk Sirinukunwattana, David RJ Snead, and Nasir M Rajpoot. "A stochastic polygons model for glandular structures in colon histology images". In: *IEEE transactions on medical imaging* 34.11 (2015), pp. 2366–2378.

[81] Korsuk Sirinukunwattana et al. "Gland segmentation in colon histology images: The glas challenge contest". In: *Medical image analysis* 35 (2017), pp. 489–502.

[82] Umberto Veronesi et al. "Sentinel-node biopsy to avoid axillary dissection in breast cancer with clinically negative lymph-nodes". In: *The Lancet* 349.9069 (1997), pp. 1864–1867.

[83] Ruqayya Awan et al. "Glandular Morphometrics for Objective Grading of Colorectal Adenocarcinoma Histology Images". In: *Scientific reports* 7.1 (2017), p. 16852.

[84] Ignacio Arganda-Carreras et al. "Crowdsourcing the creation of image segmentation algorithms for connectomics". In: *Frontiers in neuroanatomy* 9 (2015), p. 142.

[85] Albert Cardona et al. "An integrated micro-and macroarchitectural analysis of the Drosophila brain by computer-assisted serial section electron microscopy". In: *PLoS biology* 8.10 (2010), e1000502.

[86] Adam Goode et al. "OpenSlide: A vendor-neutral software foundation for digital pathology". In: *Journal of pathology informatics* 4 (2013).

[87] Adam Goode and M Satyanarayanan. "A vendor-neutral library and viewer for whole-slide images". In: *Computer Science Department, Carnegie Mellon University, Technical Report CMU-CS-08-136* (2008).

[88] Marcus D Bloice, Christof Stocker, and Andreas Holzinger. "Augmentor: An Image Augmentation Library for Machine Learning". In: *arXiv preprint arXiv:1708.04680* (2017).

[89] François Chollet et al. *Keras*. https://github.com/fchollet/keras. 2015.

[90] James Bergstra et al. "Theano: A CPU and GPU math compiler in Python". In: *Proc. 9th Python in Science Conf*. Vol. 1. 2010.

[91] Victor W Lee et al. "Debunking the 100X GPU vs. CPU myth: an evaluation of throughput computing on CPU and GPU". In: *ACM SIGARCH computer architecture news* 38.3 (2010), pp. 451–460.

[92] Marc Macenko et al. "A method for normalizing histology slides for quantitative analysis". In: *Biomedical Imaging: From Nano to Macro, 2009. ISBI'09. IEEE International Symposium on*. IEEE. 2009, pp. 1107–1110.

[93] Francesco Ciompi et al. "The importance of stain normalization in colorectal tissue classification with convolutional networks". In: *Biomedical Imaging (ISBI 2017), 2017 IEEE 14th International Symposium on*. IEEE. 2017, pp. 160–163.

[94] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[95] Santa Di Cataldo. *Overview of Machine Learning and Pattern Recognition*. Dipartimento di Automatica e Informatica Politecnico di Torino, Torino, ITALY.

[96] Barbara Finlay Alan Agresti. *Metodi statistici di base e avanzati*. 2012.

[97] Duncan Cramer Dennis Howitt. *Introduzione alla statistica per psicologia. Ediz. mylab. online*. 2014.

[98] Luigi Leone A. Paola Ercolani Alessandra Areni. *Statistica per la psicologia vol.2. Statistica inferenziale a analisi dei dati*. 2002.

[99] Minsoo Rhu et al. "vDNN: Virtualized deep neural networks for scalable, memory-efficient neural network design". In: *Microarchitecture (MICRO), 2016 49th Annual IEEE/ACM International Symposium on*. IEEE. 2016, pp. 1–13.