UNIVERSIDAD DE CHILE
FACULTAD DE MEDICINA
ESCUELA DE POSTGRADO

"Automatic Segmentation of Epithelium in Cervical Whole Slide Images to Assist Diagnosis of Cervical Intraepithelial Neoplasia"

Felipe Ignacio Miranda Ruiz

TESIS PARA OPTAR AL GRADO DE MAGISTER EN INFORMÁTICA MÉDICA.

Director de Tesis: Prof. Dr. Steffen Härtel

Co-director: Prof. Dr. Niels Grabe

2020

UNIVERSIDAD DE CHILE
FACULTAD DE MEDICINA
ESCUELA DE POSTGRADO

"Automatic Segmentation of Epithelium in Cervical Whole Slide Images to Assist Diagnosis of Cervical Intraepithelial Neoplasia"

Felipe Ignacio Miranda Ruiz

TESIS PARA OPTAR AL GRADO DE MAGISTER EN INFORMÁTICA MÉDICA.

Director de Tesis: Prof. Dr. Steffen Härtel

Co-director: Prof. Dr. Niels Grabe

2020

Acknowledgements

This work was supported financially by a DAAD (Deutscher Akademischer Austauschdienst) grant at Heidelberg University, by FONDECYT 1181823, and by the Steinbeis Transfer Center for Medical Systems Biology.

I would like to thank Dr. Niels Grabe for giving me the opportunity to be a part of the team at TIGA, and for his guidance in the development of my work. Without his support this would not have been possible. I would also like to thank the team from TIGA, Dr. Bernd Lahrmann, Alexandra Krauthoff and Liam Bartels, for their constant support and disposition.

I express my gratitude to Dr. Steffen Härtel for giving me the opportunity to work at the Centro de Informática Médica y Telemedicina (CIMT) of the Universidad de Chile during my time in the master's program, for his guidance in the development of this thesis and for believing in me in spite of my reputation as an "industrial engineer".

I thank Dr. Hartmut Dickhaus and Dr. Igor Nova for enabling my stay in Heidelberg, and for helping me with the challenges involved in moving to a new place.

I thank Dr. Nico Wentzensen from the Division of Cancer Epidemiology of the National Cancer Institute for providing the biopsies that constituted the core of this work.

I would like to thank my friends at the Universidad de Chile for making my experience there so much fun. Special mention n°1 to Marcela Aguirre for basically knowing everything, and for being my "moral" compass. Special mention n°2 to Juan José Alegría for his invaluable help during my first steps of this work.

Finally, I would like to thank my family and friends for their love and support in the difficult times we have had to live during this year 2020. I miss and love you guys deeply.

Table of Contents

Abstract

Cervical Intraepithelial Neoplasia (CIN) is a precancerous state of the cervix, and the correct diagnosis of its level of severity (CIN grade) allows to determine patient treatment and prevent invasive carcinoma. It is diagnosed from histological samples by visually estimating the percentage of epithelial width covered by neoplastic cells, although the subjective nature of this estimation has proved to cause high levels of inter-observer variability in the diagnosis. Immunohistochemical (IHC) stains have been used to increase inter-observer agreement, and to perform immune cell analysis for cervical cancer research, but the manual quantification of IHC is still prone to subjective interpretation, and unfeasible to implement on a large scale.

Digital pathology enables the digitalization of samples into Whole Slide Images (WSI), using tissue scanners that apply the principles of light microscopy. Because CIN occurs in the epithelial tissue, the first step towards a computer-assisted quantification of IHC stains would be to automatically segment the epithelium in WSIs. Deep Learning (DL) has been used in digital pathology to detect histological regions with lesions, quantify biomarkers and segment tissue types.

In digital pathology, image properties such as color, brightness, contrast or blurriness may vary based on the scanner, the tissue preparation and the technician handling the equipment. The application of DL for image analysis in this field must be robust to image variability, and deliver consistent results regardless of the origin or handling of the image. Stability Training (ST) was evaluated as a potential method to increase robustness of DL models in digital pathology. During model training, ST generates a distorted version of the training images, and penalizes differences between the prediction of the original image and the prediction of the distorted version.

This work evaluated if the application of ST could improve the DL models' robustness to image variations caused by the use of different IHC stains and different scanners, and the results obtained suggest that it makes DL models better at handling WSIs with image variations.

Three scanner models (NanoZoomer-HT, NanoZoomer-XR and NanoZoomer-S360) and three IHC stains (p16, CD3 and CD8) were used for this work. 114 WSIs stained with the p16 IHC

and scanned with the NanoZoomer-HT (p16-HT slides) were used to train a set of DL models with different weights for the stability component (α) and different distortion combinations, including a model with no ST for reference (α=0). The models were tested on six test sets (23-24 WSIs each) with different IHC stain and scanner combinations: p16-HT, p16-XR, p16-S360, CD3-HT and CD8-HT. An additional test set was used to evaluate robustness to blurriness caused by lack of lens focus (p16-HT-out of focus). Area Under the ROC Curve (AUC) was used to evaluate model performance.

The models with ST outperformed the model without ST in all test sets. All models had a similar performance on the p16-HT test set (AUC of 0.978-0.985). For p16-HT-out of focus, the model with no ST had a significantly lower performance than the best model with ST (AUC 0.876 vs 0.939). The same was observed for p16-XR (AUC 0.913 vs 0.962), p16-S360 (AUC 0.840 vs 0.962), CD3-HT (AUC 0.973 vs 0.985) and CD8-HT (AUC 0.961 vs 0.980). Distortion combination 3 (high image distortion) and α=10 were provided the best ST results.

1. Introduction

Worldwide, healthcare is undergoing a digitalization process. The use of digital medical imaging is increasing [1], making vast amounts of medical data available and easily accessible for clinical and research purposes. One clinical field that is currently at the early stages of the digitization process is clinical pathology. Here the concept of *digital pathology* is emerging, where tissue slides are converted into super-resolution Whole Slide Images (WSIs) that can be accessed and visualized from a computer. An increasing number of hospitals and research centers worldwide is adopting the use of slide scanners and digitizing vast amounts of slides into WSIs.

The growing availability of medical images is allowing researchers and data scientists to develop new technologies involving sophisticated image analysis tools [2], [3]. In this context, Deep Learning (DL) has emerged as a powerful methodology to analyze and generate relevant information from images, with applications in multiple medical fields [4]. In digital pathology, it is becoming a standard tool for tissue segmentation, classification and diagnosis, mostly in cancer research. However, most of the current research is performed on homogeneous image sets, from tissue samples prepared with the same stains, and digitized with the same scanner owned by the research team. Given the existence of multiple different scanners and stains, the scope of digital pathology has a more heterogeneous nature: images will not necessarily look alike. Therefore, Deep Learning models must be robust to these image variations in order to be applicable in real clinical contexts.

This thesis project is set at the cross-road between medical imaging, Deep Learning and clinical research. It aims to contribute to the field by applying Deep Learning in the research of new tools to diagnose Cervical Intraepithelial Neoplasia, a precancerous state of the cervix. For this purpose, methods focused on increasing predictive robustness and tolerance to image variations are applied.

1.1. Thesis structure

This thesis is structured as follows: Chapter 1 describes the clinical relevance of *Cervical Intraepithelial Neoplasia* and the concepts of *digital pathology* and *Deep Learning*, and connects these with the motivation and objectives of the project. Chapter 2 summarizes the current advances in the application of Deep Learning in digital pathology, and describes approaches that have been applied to increase predictive robustness in different scenarios. Chapter 3 describes the materials and methods that were used during the execution of this work, and its subsection general approach describes the sequence of activities that were performed, as well as the materials and methods involved in each activity. Chapter 4 describes the main results obtained, in coherence with the project's objectives. These results are divided into (1) background removal and data preparation, (2) base architecture selection and hyper-parameter fine-tuning and (3) Stability Training and evaluation of robustness. Chapter 5 discusses the interpretations, implications and limitations of the results and provides an answer to the hypothesis of this work. Chapter 6 concludes this work, with an overall view of the key findings and the relevance of this work in the field.

1.2. Cervical Intraepithelial Neoplasia (CIN)

The following section provides a characterization of Cervical Intraepithelial Neoplasia (CIN) as a precancerous condition of the cervix, from an epidemiological and clinical perspective. The level of severity of CIN is a relevant aspect when evaluating the probability of a patient of developing invasive cervical cancer, and its diagnosis is the basis to select the correct treatment. The histopathological diagnostic process of CIN is described as well, along with its technical limitations. This work makes a first step towards addressing these limitations through the use of Deep Learning.

1.2.1. Cervical Cancer and CIN

Cervical cancer is one of the most common forms of cancer in women. It ranks 4th worldwide in both incidence and mortality among females. In 2018 there were 570.000 new cases and 311.000 thousand deaths related to cervical cancer. This is the leading cause of cancer death in many lower income countries [5].

The most frequent forms of cervical cancer are squamous cell carcinoma (SCC) and adenocarcinoma (AC) with 80-90% and 10-15% of the total number of cases respectively. Both of them are associated with the Human Papilloma Virus (HPV), which is present in 99.7% of all cervical cancer cases [6]. The natural history of the disease can be described in the following stages [7]:

1. HPV transmission and infection of the metaplastic epithelium: HPV is transmitted by skin-to-skin or mucosa-to-mucosa contact. The virus reaches and infects germinal cells in the basal layer of the epithelium, usually near the transformation zone[1] (Fig. 1.1).
2. Viral persistence: Some HPV infections are more persistent, making them harder to eradicate by the body's immune system. The extended duration of these infections causes the proliferation of abnormal cells.
3. Progression of persistently infected epithelium to cervical pre-cancer: As the proliferation of abnormal cells persists, almost the whole thickness of the epithelium becomes occupied by abnormal cell tissue. This is considered a cervical pre-cancer contained within the epithelium, also called CIN.
4. Invasion through the basement membrane of the epithelium: Eventually, the abnormal cell tissue invades the cervical stromal tissue. This is diagnosed as invasive cervical cancer.

[1] Zone of the cervix where the columnar epithelium collides with metaplastic squamous epithelium. It is vulnerable to infection because of the high levels of cell proliferation in the area.

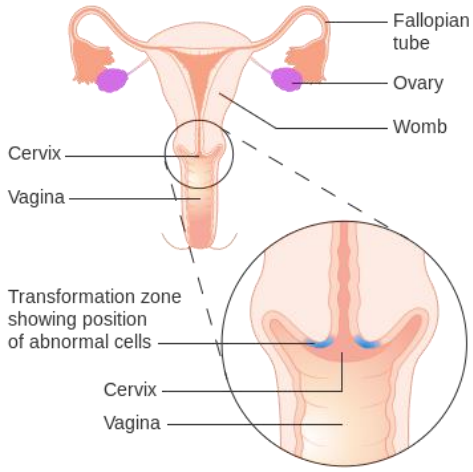


Fig. 1.1: Diagram showing the transformation zone of the cervix[2]. In this region, the columnar epithelium covering the endocervical canal is replaced by squamous epithelium. This is the most likely area for cells to become cancerous.

Cervical cancer usually presents a slow progression, taking as much as 10 years to evolve from an epithelial lesion to invasive carcinoma. For this reason, it allows the opportunity for prevention and early detection. However, the existence of prevention programs is required for early screening, early stage diagnosis and treatment of the disease. Countries without such programs present high incidence and mortality rates, as is the case of countries in sub-Saharan Africa, South East Asia, South America and the Caribbean [8].

CIN is the condition where a layer of HPV infected cells begins to grow from the basal levels of the epithelium to the top levels, as seen in Fig. 1.2. It corresponds to the stages 2 and 3 within the natural history of cervical cancer. The diagnosis of CIN grade (the level of severity) will determine the treatment for patients [9], so diagnostic accuracy is essential to avoid under or overtreatment. The correct treatment of CIN can prevent a progression to invasive carcinoma.

[2] "Diagram showing the transformation zone of the cervix" by Cancer Research UK is licensed under CC BY-SA 4.0. https://commons.wikimedia.org/w/index.php?curid=34334321

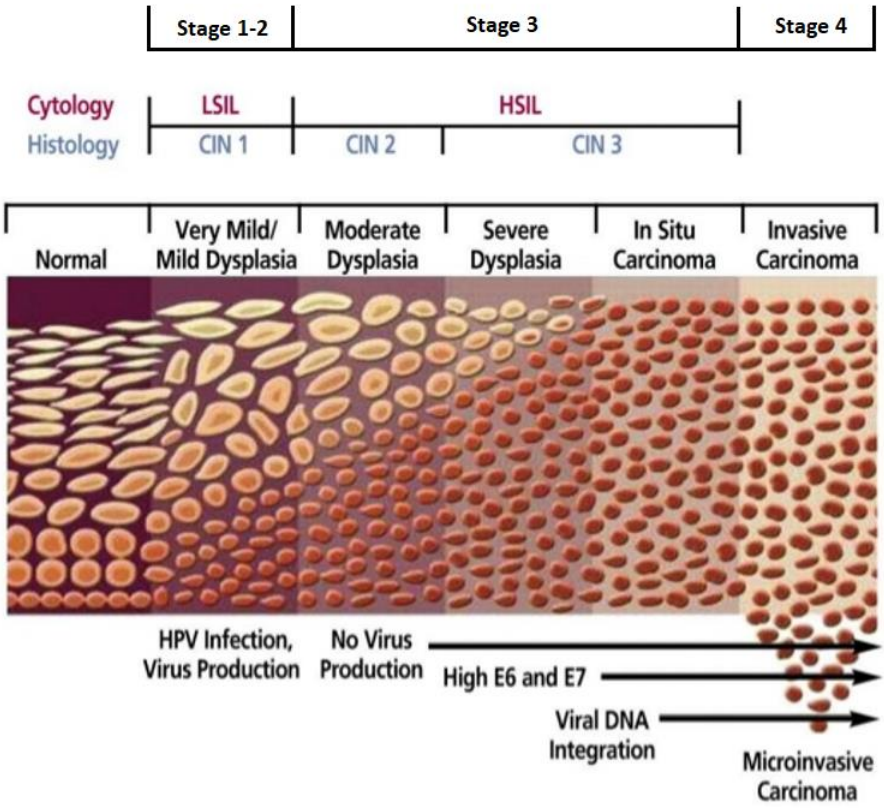


Fig. 1.2: Progression from a benign cervical lesion to invasive cervical cancer [10]. Neoplastic cells (small dark cells at the bottom of the image) begin to cover a larger proportion of the epithelial width as these proliferate from the basal layer of the epithelium. Eventually these cells can cover the whole epithelial width and invade into the stromal tissue.

CIN is considered a pre-cancerous state of the cervix. Although the presence of CIN does not necessarily imply that the patient will have cervical cancer, the CIN grade is correlated to the probability of progression to invasive carcinoma. The level of severity of this disease is divided into grades 1, 2 and 3, which are diagnosed based on the proportion of the epithelial width that is used by abnormal cells:

- CIN1: abnormal cells in the base of the epithelium cover up to 1/3 of the epithelial width. This grade is associated with low risk of developing cervical cancer. It usually regresses naturally and does not require treatment, even though 10-20% of the cases progress to CIN3 or worse [11].

- CIN2: abnormal cells in the base of the epithelium cover up to 2/3 of the epithelial width. This grade is associated with a mid-high risk of developing cervical cancer, and presents a 25% chance of progressing to CIN3 or worse [11]. The CIN grade does not necessarily progress linearly from CIN1 to CIN2, since both are generally caused by different HPV strains. CIN2 is generally caused by HPV 16 or 18, known to be more persistent strains and more correlated to cervical cancer.
- CIN3: abnormal cells cover between 2/3 and the whole width of the epithelium. This grade is associated with a high risk of cervical cancer, and presents a 12% chance to progress to invasive carcinoma [11]. It is considered a precancerous state of the uterine cervix [12].

The CIN diagnosis is usually performed when a patient presents abnormal or unclear results in a screening by cytology testing (i.e. Pap smear), as a part of the cervical triage [13].

1.2.2. Diagnosis of CIN and associated limitations

For the diagnosis of CIN, a tissue sample is taken from the patient during a colposcopy, from which tissue slides are generated (i.e. slices of the tissue). A pathologist performs a visual interpretation of the slides to evaluate the percentage of the epithelium width presenting abnormal cells. With that information the CIN grade can be determined, which is then used to determine the appropriate treatment and clinical management of the condition. However, this estimation is a qualitative assessment, performed visually by a pathologist. As a consequence, there is a high inter-observer variability in the interpretation of results, particularly in the distinction between CIN1 and CIN2 [14], [15]. The problem with this variability is that the treatment for both grades is different, so a wrong diagnosis can lead to treatments that are not suited for the condition. Results from a study regarding inter-observer variability in the diagnosis of CIN grade can be observed in Table 1.1. Here, a set of biopsies with their respective diagnoses were analyzed by a panel of experts. Results showed that only 38.2% of the CIN1 diagnoses (539 out of 1.411), 38.0% of the CIN2 diagnoses (872 out of 2.295) and 68.0% of the CIN3 diagnoses (1.231 out of 1.811) from the clinical community were confirmed as such by the panel of experts.

		Expert Panel Diagnosis					
		Negative	CIN1	CIN2	CIN3	Cancer	Total
Community Diagnosis	Negative	**507**	22	11	1	0	541
	CIN1	736	**539**	115	17	4	1411
	CIN2	388	602	**872**	429	4	2295
	CIN3	104	68	391	**1231**	17	1811
	Cancer	16	1	3	35	**159**	214
	Total	1751	1232	1392	1713	184	**6272**

Table 1.1. High diagnostic inter-observer variability between original diagnoses (Community Diagnosis) and expert panel diagnoses of the same images (Expert Panel Diagnosis). As an example, of the 1.411 cases diagnosed as CIN1 by the treating pathologists (green), only 539 were confirmed as such by the expert panel (red), equivalent to 38.2% of the cases. High levels of inter-observer variability can also be observed for CIN2 (872/2.295) and CIN3 (1.231/1.811) Source: [15].

Immunohistochemistry (IHC) is applied in different forms of cancer for diagnosis, stratification, staging and evaluation of response to therapy [16]. It consists in the application of antibodies to detect specific antigens in tissue. Through this process, tissue samples can be stained to highlight relevant tissue for an easier visual interpretation. Regarding cervical cancer, the p16 IHC has proven to improve inter-observer agreement in the diagnosis of CIN [17]. $P16^{INK4a}$ is a protein that was discovered to have an important role in cell cycle regulation [18]. It binds to a specific cyclin-dependent kinase called CDK4, inhibiting or slowing down its role in cell proliferation. In turn, CDK4 regulates the expression of $P16^{INK4a}$ through a process called Retinoblastoma phosphorylation (pRb). The Human Papilloma Virus (HPV) inhibits the process of pRb, resulting in an over activation of the cell duplication cycle that causes an over-expression of $P16^{INK4a}$ as a byproduct [19]. HPV also degrades a protein called p53, responsible for regulating cell death (apoptosis). The combination of pRb inhibition and p53 degradation causes epithelial cells to become neoplastic, and can eventually cause the development of CIN. The over-expression of $P16^{INK4a}$ makes it a suitable surrogate marker for HPV-related neoplasia [19]. Visually, the p16 IHC highlights the protein as a brown stain in neoplastic cells, as can be observed in Fig. 1.3.

Immune cell analysis is another field where IHC is being applied. Immune cells have been reported as biomarkers to predict patient outcome and response to treatment [20]. T cells have shown higher density in stromal regions where there is inflammation of tissue caused by the presence of tumors. In 2006, Galon et al. discovered that the presence of CD3+ and CD8+ T cells near colorectal tumors could be used to predict patient survival [21]. Further research has also shown the presence of higher densities of FOXP3+ and CD8+ T cells in colorectal cancer [22]–[24].

With regards to cervical cancer, research suggests that FOXP3+ might also be a relevant biomarker for patient stratification (i.e. classification of patients based on cancer type), though large cohort studies are described as required to validate this hypothesis [20]. Immune cell analysis has also been used to evaluate patient response to experimental treatments for cervical cancer, as well as to analyze immune response to chemotherapy [25], [26]. In the case of CIN, immune cell analysis was performed on a small cohort to evaluate the relation between immune responses and CIN grade progression [27]. Examples like these suggest that immune cell analysis in cervical cancer is currently an open field, with potential to assist in the research for new treatments. It could also be a useful method to stratify patients based on their cancer type, to evaluate response to treatment and to predict patient outcome. The use of IHC enables the visualization of immune cells, such as CD3+ and CD8+ T cells (using CD3 and CD8 biomarkers, respectively), as can be as can be observed in Fig. 1.3.

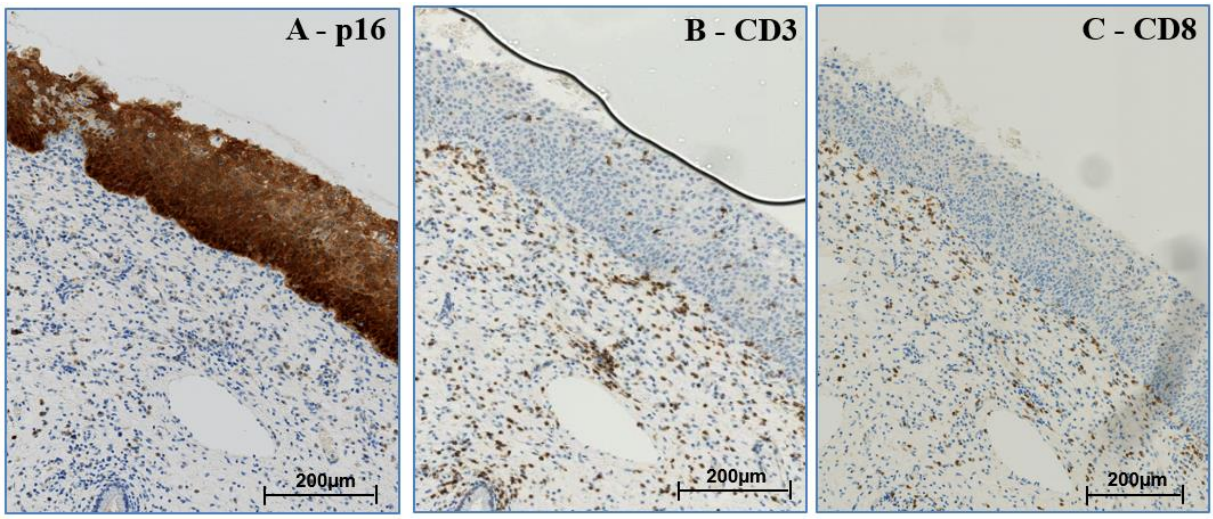


Fig. 1.3. Example of IHC stains applied to cervical tissue. (A) p16-stained tissue. Neoplastic cells (marked in brown) can be observed covering the whole width of the epithelium. (B) CD3-stained tissue. CD3+ T cells (marked in brown) can be observed within the stromal region, surrounding the epithelium. (C) CD8-stained tissue. CD8+ T cells (marked in brown) can be observed within the stromal region, surrounding the epithelium as well. These WSIs were taken at 40x resolution (1 pixel = 0.2276µm)

So far, p16 has been successfully used to assist in the diagnosis of CIN. However, the visual estimation of neoplastic tissue width in the epithelium is still subject to variability. On the other hand, the potential use of immune cell analysis for patient stratification, evaluation of response to treatment and prediction of patient survival rate is currently under research. This kind of analysis requires an automated approach to quantify the presence of immune cells (such as CD3+ and CD8+, as shown in Fig 1.3), since a manual calculation is prone to human error, time consuming and overall infeasible on a large scale, both for research and for an eventual clinical application (if these methods are validated). For any of these automated processes, the automatic identification of epithelium within tissue samples can allow faster image analysis. All further calculations can then be applied on this region and its surroundings.

1.3. Digital Pathology

In the past, radiologists relied on the capture of analog images, generated by the exposition of special films to certain radiations such as X-ray. These images had to be manually delivered to the physician for his interpretation. Now, digital technology allows images to be captured, stored, edited and visualized online. The digitalization of radiology opened the field to new possibilities, making diagnoses faster and more precise, facilitating case management and improving image quality [28].

In the context of the digitalization of healthcare, digital pathology is emerging as a new approach to conventional pathology. Conventional cytopathology and histopathology are based on the preparation of tissue slides and their visualization on light microscopes. Pathologists need the physical slides at their disposal, and perform their diagnoses based on the visual interpretation of the sample. The basis of digital pathology is the digitalization of these slides into Whole Slide Images (WSIs) by means of a tissue scanner, which applies the principles of light microscopy to generate super-resolution images from the slides. These WSIs can then be edited, visualized and annotated by means of specialized visualization software (see Fig. 1.4) [29].

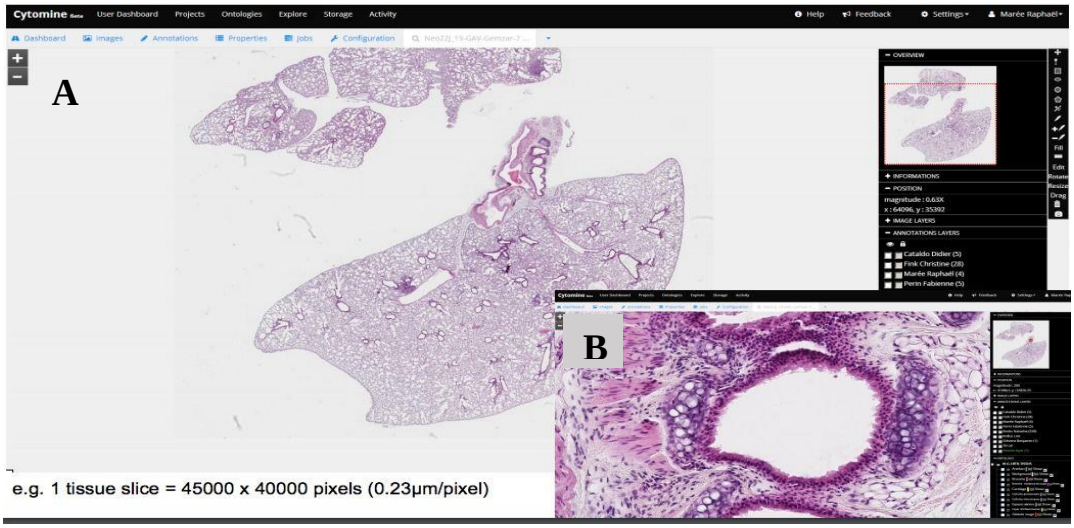


Fig. 1.4. Example of the Cytomine interface for the visualization of WSIs. Cytomine is an open source slide viewer. Two separate windows can be observed in this figure. Window *A* shows the WSIs with the maximum zoom out. Window *B* shows a zoom in on a specific region. Both windows present a menu on the right side, containing a map of the WSI and buttons to toggle annotations. Source: [30]

Digital pathology has enabled multiple clinical applications. The possibility of remotely visualizing WSIs by means of a personal computer has allowed pathologists to perform telepathology, inter consultations and the visualization of samples during tumor boards (inter-specialty reunions to discuss cancer-related clinical cases) [29]. The digitalization of tissue slides has also allowed for the long-term storage and retrieval of slides, solving the need to store the physical slides and to have these locally available [31]. Through the visualization software, pathologists can also annotate and label regions within the WSIs, which are also stored and can be shared with other practitioners. Research in the field also suggests that there is a good diagnostic concordance between WSIs and conventional glass slides [32]–[34].

There are still some limitations associated with the adoption of WSI technology. The switch from conventional to digital pathology requires practitioners to become acquainted with the new technology. On this point, some users find the visualization software counterintuitive to use, and argue that lag due to connection problems makes the process inefficient. These technical difficulties may limit the daily use of digital pathology for routine, high-volume diagnostics [31]. Another limitation is the (always present) economic factor. Currently, high-throughput slide scanners are quite expensive, bordering USD 200.000. However, this cost is not so distant from the cost of other medical equipment, and a 12% compound annual growth rate is estimated for the WSI market in the 2010-2027 period [35]. Challenges as the ones described are not uncommon when new technology is introduced in the medical field. Usually, its adoption depends strongly on the disposition of the community to make the leap. The advantages associated with slide digitization cannot be argued, and its limitations will most likely be overcome as the technical limitations are addressed and as new generations recognize its potential.

1.4. Brief history of Deep Learning

With the adoption of digital pathology, vast amounts of WSIs are being captured and stored. Not only are the images stored, but also relevant clinical information associated with the images, such as labelled annotations and overall diagnoses. As the reader might already intuit, this increasing availability of medical data serves as a golden opportunity for researchers to harness its potential, and develop new technologies aimed at improving the practice of medicine. This

section introduces the concept of Deep Learning, a machine learning methodology that has been showing remarkable practical applications in multiple fields, including the biological and clinical fields. But, what is the idea behind Deep Learning?

In the field of machine learning, the idea of emulating the brain functioning to solve complex problems served as inspiration to develop the first artificial neural networks. The origins of this idea can be traced back to the 40's, with a paper from McCulloch and Pitts proposing a mathematical model for the workings of neurons, and describing a neural network built using electrical circuits [36]. However, complex neural networks required a processing capacity that was not available at the time. More than a decade later, in 1959, Widrow and Hoff from Stanford created the first neural network to be applied on a real world problem: eliminating echoes on phone lines[3] [37]. After a silent period caused by disappointment due to high expectations, an algorithm called backpropagation emerged as a new method to train neural networks. A practical demonstration of the algorithm was performed by researcher Yann LeCun in 1989, using the network to read handwritten digits.

In 1998, Yann LeCun published a ground-breaking paper describing the use of Convolutional Neural Networks (CNNs) for the recognition of handwritten characters in documents [38]. Convolutions allowed to extract relevant, complex patterns from the data without an excessive computational cost. Even though CNNs had been used in the late 70's by Japanese computer scientist Kunihiko Fukushima [39], the processing capabilities and the development of GPUs during the 90's allowed for more complex models to be developed. These new models had a higher number of layers, making them deeper than their predecessors (hence the name Deep Learning). LeCun's paper combined CNNs, backpropagation and gradient descent (an optimization algorithm), which constitute the base of modern Deep Learning. However, it was not until 2012 that research in the field exploded, with the publication of Alex Krizhevsky's AlexNet [40]. This new network was able to achieve a remarkably high performance on an (at the time) infamously difficult and popular dataset called ImageNet (see Fig. 1.5), causing a spike in the interest for Deep Learning by the community.

From that point on, the development of Deep Learning applications has been growing non-stop.

[3] This network, called MADALINE, is still in commercial use!

Programs can now recognize objects in real time videos [41], beat the world champion in the game of Go [42], drive cars [43], suggest movies on Netflix, make faces younger or older, or even generate realistic human faces [44], classify skin lesions with dermatologist precision [45], act as visual assistants, process natural language, colorize black and white photos [46] and hundreds of other applications that pass unnoticed in daily life.

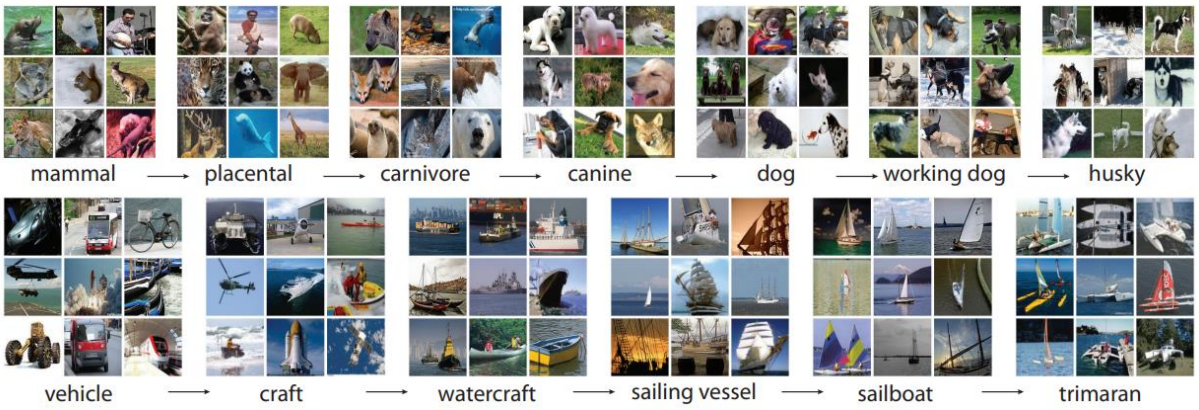


Fig. 1.5. Example of "root to leaf" branches from ImageNet database. The database is an ontology of images, manually labelled with their hierarchical classes. Currently it contains more than 14 million images and 20.000 classes. The arrows describe the hierarchical relation between image classes, so an image corresponding to a "husky" can be traced to all parent classes up to "mammal". The same with an image corresponding to a "trimaran", which can be traced to "vehicle". This dataset became widely used in Deep Learning research. Source: [47]

Currently, Deep Learning has taken over the field of image analysis, including object classification, image segmentation and even image generation. There are multiple examples of Deep Learning applications in the biological and clinical fields. Some of these, in the field of digital pathology, are described later in this work. Deep Learning is silently permeating in all aspects of our society without us being even aware of it, and for now, it seems like it is not going to stop. As Wang et al. elegantly states it, "this period is the era to witness the blooming of deep learning research" [48].

1.5. Motivation for this work

The following work is framed within an ongoing collaborative research program between the Tissue Image and Analysis Center[4] (TIGA) and the US National Cancer institute (NCI). The goal of this work was to use Deep Learning to automatically segment the epithelium in histopathological cervical WSIs. The automatic segmentation will serve as an input to extract quantitative indicators from the WSIs and evaluate whether or not these can reliably assist in the diagnosis of Cervical Intraepithelial Neoplasia (CIN).

One of the main challenges of Deep Learning in digital pathology is the heterogeneity that exists in terms of sample preparation and scanning quality. During sample preparation, different staining methods result in different samples in terms of color and texture. At the same time, different scanner models usually scan in slightly different colors and apply different focal points, even if these come from the same manufacturer. As a consequence, WSIs will often present certain variations in color, contrast, brightness, and blurriness. In order for a Deep Learning-based solution to be practical in the real world it needs to be robust to such variations, which is usually not the case when a Deep Learning model is only trained and tested with WSIs from the same origin.

The purpose of this work was not only to segment epithelium, but to do so with a high level of robustness. This means being able to obtain a high segmentation performance for WSIs with different image properties. A technique called "Stability Training" [49], which aims to improve the robustness of Deep Learning models, was used to achieve this goal. To the author's knowledge, this technique has not been applied in digital pathology so far.

As input for this research, the NCI has collected more than two hundred histopathological cervical tissue samples. Samples were prepared by TIGA using three different immunohistochemical (IHC) stains (p16, CD3 and CD8), each one marking different elements that may be relevant for the generation of quantitative indicators to assist in the diagnosis of CIN. The prepared samples were scanned using three different tissue scanner models provided by the Japanese company Hamamatsu.

[4] Dr. Niels Grabe, co-director of this work, is scientific director of TIGA.

1.6. Hypothesis and Objectives

The following hypothesis was assessed by the execution of this work:

The models trained with Stability Training will outperform the model trained without Stability Training in the segmentation of epithelium in WSIs from different scanners and WSIs with different IHC stains.

The main objective of this work was to evaluate the robustness of Deep Learning models trained with Stability Training in the context of segmenting epithelium from histopathological cervical images from different scanners and WSIs with different IHC stains.

The following specific objectives were proposed:

(a) Preprocess the image sets into datasets for Deep Learning model training and evaluation;
(b) Select the optimal base architecture and hyper-parameter values to be used during Stability Training;
(c) Train a set of Deep Learning models using different levels of Stability Training on a dataset of WSIs with the same IHC staining and from the same scanner;
(d) Compare the segmentation performance of these models with a model trained without Stability Training on WSIs from different tissue scanners and with different IHC stains.

2. Related work

The purpose of chapter 2 is to briefly put this work in context with the current advances in the application of Deep Learning in digital pathology. The first section of this chapter provides a glimpse of the major advances in the application of Deep Learning in the field. The number of research publications is increasing significantly, which is why it would be impossible to fully describe it in this work. Nonetheless, this section describes a survey of Deep Learning applications in digital pathology. The current applications cover multiple pathologies (especially cancer-related), multiple sample preparations and multiple use cases.

The second section of this chapter addresses the challenges associated with heterogeneous image data in the field of digital pathology, and describes some of the solutions that have been proposed in order to increase robustness of Deep Learning applications. Robustness emerges as a relevant element to consider in a world where WSIs can come from different locations and have different preparations. Training for robustness can allow researchers to exploit the vast amounts of data available, and not be restricted to self-generated, self-curated image sets.

2.1. Applications of Deep Learning in Digital Pathology

Deep Learning (DL) has emerged as a powerful tool for image classification and pattern recognition using WSIs. In its survey, Litjens et al. [4] describes the main scientific developments within the field, including disease classification at lesion or WSI level, tissue segmentation and quantification of biomarkers. According to this survey, the most frequent application of Deep Learning is for nucleus segmentation and classification, and for mitosis detection, both using H&E staining. Another frequent use case is the segmentation of colon glands.

An interesting study was found, which used Convolutional Neural Networks (CNNs) to segment epithelium and stroma in breast and colorectal tissue, as can be seen in Fig. 2.1 [50]. In this work, the focus was on evaluating different methods for image preprocessing before training with the CNN, as well as using different methods to classify cell nuclei as complementary

information for the CNN. Another interesting use case was the segmentation of cytoplasm and nuclei in cervical Pap smears [51].

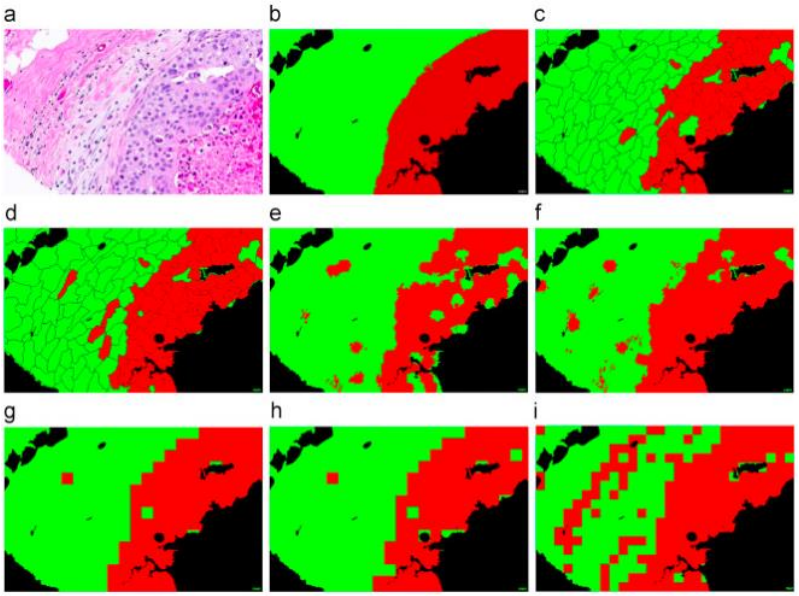


Fig 2.1. Segmentation of epithelial and stromal regions on a tissue image (*a*) from research by Xu et al [50]. Image *b* corresponds to the ground truth for the epithelial region (red). Images *c* to *i* show the segmentation results of different CNNs, where the red region is the predicted epithelium.

Within the more recent publications, software based on Deep Learning has been applied in the detection of injury in gastric histological WSIs [52]. Clinical decision support systems have been created as well, to screen for negative samples in breast tissue [53] and to provide relevant indicators in thyroid cytopathology [54]. The relevance of these projects is that they provide a glimpse of the future of pathology: a future where pathologists can gain access to new, quantitative, automatically extracted information to support their diagnostic processes. An enormous potential for improvement exists in this field (and in medicine in general) with the digitalization of data and the application of innovative techniques for data analysis.

With the growing availability of tissue scanners worldwide, vast amounts of image data are becoming increasingly available. As a consequence, publications in the field of digital pathology and Deep Learning is constantly expanding. New challenges emerge, however, as research

teams usually use images that are prepared with the same stains and scanned with the same scanners. The application of Deep Learning in digital pathology comes now with the challenge of making Deep Learning robust to images from different origins and with different preparations.

2.2. Training for robustness in Digital Pathology

In the field of digital pathology, variations in image properties (i.e. heterogeneity) can be caused by the type of tissue scanner, the type of stain used for the preparation of slides or even by changing the person handling the equipment. Heterogeneity can be found even within the same WSI, in the form of blurry and out of focus regions, folded tissue, uneven staining and other artifacts. In this context, robustness in Deep Learning is introduced as a measure of the ability of a Deep Learning model to sustain a high level of performance when analyzing images with properties different than those of the images used for training. Those differences may come in the form of different image quality, size, color, contrast or blurriness, and a robust Deep Learning application should be able to perform regardless of these differences.

Regarding within-WSI heterogeneity, the identification of cell types can be a challenge. Even in tissues with the same type of cells, the morphology of each cell can vary. A common reason is that the slice extracted from the tissue to prepare a slide might cut cells in different points of the Z axis. The focus points might also be slightly different in different regions of the WSI. In this context, robustness to cell morphology is introduced. Zhang et al. [55] proposes a cell segmentation method based on Active Contours and feature extraction to obtain relevant features to describe each cell. Then, an image retrieval method is used to identify each cell class from an existing dataset composed of half a million cells. This method attempts to increase robustness to variations in cell shape and size (see Fig. 2.2). Xing et al. [56] provides a review of multiple methods to improve cell detection and segmentation. The focus is on image processing methods such as intensity thresholding for cell recognition, Active Contours and noise elimination among others, though declares that robust cell detection is still an open challenge.

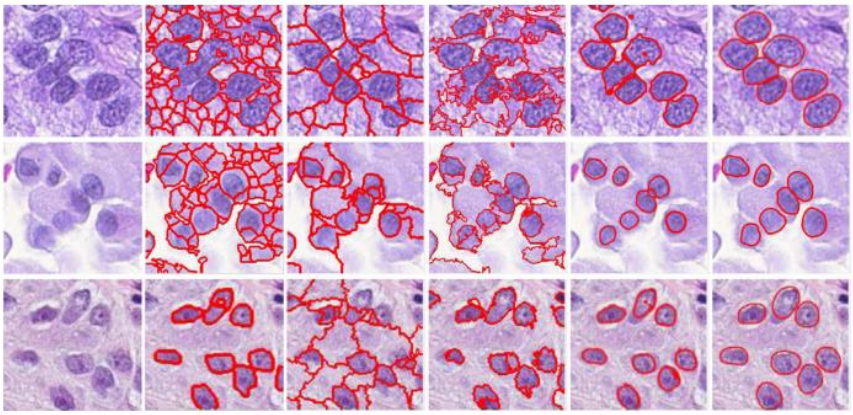


Fig. 2.2. Different nuclei segmentation methods from research by Zhang et al [55]. The study compared their new nuclei segmentation method (last column) with other methods available in the literature. The authors described their method as the best performing. Visually, the last column seems to segment nuclei more effectively.

Therrien et al. [57] proposes using image sets generated by different institutions to train Deep Learning models for robustness. The idea was to evaluate if the model performance on a test dataset X improved when using different combinations of datasets Y and Z for training. According to their results, performance improves when using both datasets instead of only one. Hosseini et al. [58] generated a digital pathology database with 18 thousand patches from 100 WSIs. The images come from different tissue types across different organs, and each image is annotated with multiple hierarchical tissue types (see Fig. 2.3). The purpose of this database is to promote research on generalized models which are able to process different types of tissue.

Veeling et al. [59] uses a Deep Learning method called "group equivariant CNNs" (G-CNNs) to improve model performance in the classification of tiles (i.e. small sections of the WSI) under reflection and rotation perturbations. These G-CNNs replace traditional convolutional layers with G-convolutional layers, designed to handle such perturbations. Balkenhol et al. [60] proposes a color transformation from RGB to H&E color space (color space specific to the hematoxylin and eosin stain, H&E), a stochastic augmentation of the new color space and a subsequent deconvolution into RGB, in order to improve the models robustness to variations associated with the origin of the WSIs. The models trained by this method are then tested in images from different institutions which present variations in the H&E coloration.

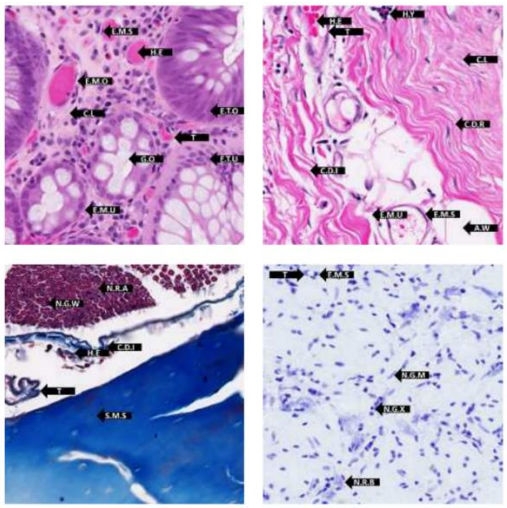


Fig. 2.3. Four examples of tissue patches from the image database generated by Hosseini et al [58]. A total of 18.000 patches were generated from 100 WSIs. Each patch has multiple annotations in the form of arrows that mark specific tissue types with a letter code. The purpose of this database is to enable research on generalized machine learning models that can classify tissue elements and regions from different organs.

3. Materials and Methods

The following chapter describes the materials and methods that were used during the execution of this work. The most relevant pieces of hardware were the tissue scanners and the high performance desktop PC. Software programs were used for WSI visualization and annotation, and for image analysis. Python libraries were used for the application of Deep Learning and for the visualization of results. The image set and the methods to obtain it are described as well. The *image preprocessing for dataset generation* section describes the steps involved in the processing of the image set into a dataset that allows the application of Deep Learning.

The *Deep Learning* section describes the basics of neural networks and Convolutional Neural Networks (CNNs), the fundamentals of Deep Learning. The concepts of loss functions and optimizers are introduced, as well as the hyper-parameters that describe their functioning. The *Stability Training for robustness* section describes a technique called Stability Training, which allows to improve the robustness of CNNs. The *Area Under the ROC Curve (AUC)* section describes the concepts of Receiver Operating Characteristic (ROC) curve and AUC, a metric commonly used to evaluate prediction performance in machine learning. The *General approach* section provides a sequential description of the activities that were performed throughout the project and the materials and methods involved in each activity.

3.1. General approach

The first step of this work consisted in gathering an *image set* of histopathological cervical WSIs with an annotated epithelium (refer to section 3.3). Some of the WSIs were already available and others had to be generated by scanning the slides with different tissue scanners (refer to section 3.2).

The second step consisted in processing the image set into a dataset, which involved an exploratory analysis of the WSIs to select the best criteria to remove the background from the images (refer to section 3.4).

The third step consisted in selecting the optimal Deep Learning base architecture (refer to section 3.5.3) and fine-tuning three hyper-parameters involved in the training process of the Deep Learning models: *learning rate*, *momentum* and *decay* (refer to section 3.5.1).

The fourth step consisted in training a set of Deep Learning models using Stability Training, with different *distortion combinations* and values for α (i.e. the weight of the stability component, or stability component weight) (refer to section 3.6).

The fifth and final step consisted in comparing the segmentation performance of the Deep Learning models trained with Stability Training with the performance of a model trained without Stability Training, using the Area Under the ROC Curve (AUC) as a performance metric (refer to section 3.7).

3.2. Hardware & software

The following section describes the hardware and software used during the execution of this project. This includes the tissue scanners used to digitize the tissue slides into WSIs, as well as the PC and applications that were used for image processing and for the implementation of the Deep Learning algorithms.

Tissue Scanners

TIGA has a collaborative agreement with Hamamatsu, a Japanese manufacturer of optical sensors and other related devices used for scientific and medical use. In this context, Hamamatsu has provided TIGA with three tissue scanner models for their use in digital pathology research. The NanoZoomer-HT is the older model available, the NanoZoomer-XR is a newer version and the NanoZoomer S360 is the newest model from the company (Fig. 3.1). The technology used for tissue scanning differs between each device, therefore rendering images with different quality and color, as described in section 3.3.

Fig. 3.1. Hamamatsu NanoZoomer S360 from the lab at TIGA. One of the scanner models used for this work.

Computer specifications

The activities involved in this project were executed using a high-performance desktop PC available in the TIGA lab. Table 2.1 describes the PC's hardware specifications, as well as the software used for visualization and image analysis, and the most relevant python libraries used for image processing and model training.

CPU	Intel Core i7-4790K @ 4.00GHz
RAM	32GB
Graphics card	NVIDIA GeForce GTX 1070 8GB
WSI visualization/annotation software	NDP.view2
Image analysis software	ImageJ
Python libraries	Openslide-python=1.1.1 Tensorflow=1.2 openCV=4.1

Table 3.1: PC hardware specifications and software used in this work

3.3. Image set

The WSIs used for this work were obtained through a collaborative work between TIGA and the Division of Cancer Epidemiology and Genetics of the National Cancer Institute (DCEG-NCI), with the purpose of researching new methods to generate quantitative indicators for the diagnosis of CIN using digital pathology and image processing. The original tissue samples were kindly provided by Dr. N. Wentzensen (DCEG-NCI). These samples were processed into glass slides by TIGA using the p16, CD3 and CD8 IHC stains. The slides were digitalized into WSIs using the scanners available at TIGA, which were provided by the Japanese company Hamamatsu. The annotation of epithelium within the WSIs was performed by researchers at DCEG-NCI using the software NDP.view2 from Hamamatsu.

Fig. 3.2 describes how the annotation of a WSI is performed using the scanners from Hamamatsu. The software NDP.view2 allows the user to visualize these high resolution WSIs in a similar way as google maps works, by loading regions of the image as needed instead of loading the whole image at once[5]. The user can then create one or many annotations by selecting a shape, such as a rectangle, an ellipse or just a freehand annotation like the one from Fig. 3.2, and marking the region(s) of interest with the cursor. These annotations are then saved in .ndpa format (a specific annotation format for WSIs scanned with Hamamatsu scanners), which stores the coordinates of the points that constitute the annotations, relative to a coordinate system that depends on the height and width of the WSI (in pixels).

[5] NDP.view2 is only compatible with WSIs that come in the .ndpi file format, which is exclusive for WSIs generated by scanners from Hamamatsu.

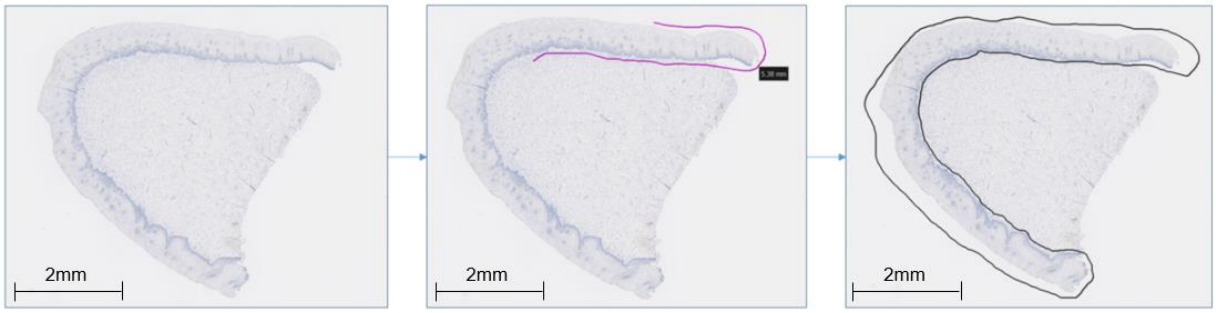


Fig. 3.2. Example of a freehand annotation of the epithelium in a cervical histopathological WSI stained with p16. The user can freely scroll the cursor and create a closed region, which is then stored as a list of points describing its shape. This image corresponds to an .ndpi file which is visualized with the software NDP.view2.

In accordance with the objectives of this work, the image set is composed of six categories of WSIs. The differences between each can be observed in Fig. 3.3. All WSIs were scanned in a 40x resolution (1 pixel = 0.2276μm).

1. *p16-stained WSIs scanned in-focus with the NanoZoomer-HT:* A total of 138 WSIs from this category were used. All WSIs had been scanned and annotated years before the beginning of this work.
2. *p16-stained WSIs scanned intentionally out-of-focus with the NanoZoomer-HT:* A total of 24 WSIs from this category were used. The slides used were the same as the ones used for the previous category. They were scanned specifically for this work, and were annotated using the same annotations of the WSIs from the previous category. In order to cause blur in the WSIs, an intentional defocus of approximately 0.5μm was applied over the automatically suggested optimal focus points during the digitization process.
3. *p16-stained WSIs scanned with the NanoZoomer-XR:* A total of 24 WSIs from this category were used. They were scanned specifically for this work as well, using the same 24 slides that were used in the previous category and the same annotations.
4. *p16-stained WSIs scanned with the NanoZoomer S360:* A total of 23 WSIs from this category were used. They were scanned specifically for this work using the same slides that were used in the previous category (except for one WSI that had to be removed from the dataset). The same annotations as in the previous category were used.

5. *CD3-stained WSIs scanned with the NanoZoomer-HT:* A total of 24 WSIs from this category were used. The slides were prepared with the CD8 stain, scanned and annotated before the beginning of this work. The slides used in this category were prepared from different slices from the same tissue samples used in the other categories, so the WSIs from this category have some minor morphological differences with the others.
6. *CD8-stained WSIs scanned with the NanoZoomer-HT:* A total of 24 WSIs from this category were used. This category has the same characteristics as the CD3-stained category, except for the use of CD8 for staining.

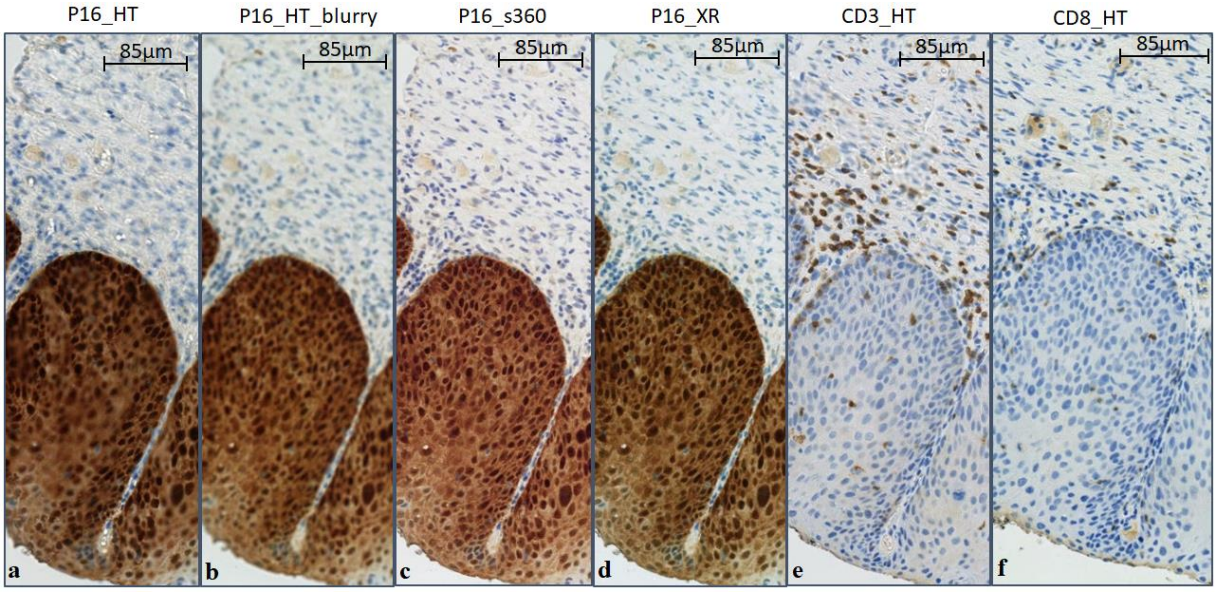


Fig. 3.3. Example of WSIs from the six categories. The most notable visual difference is the fully brown epithelium in the WSIs stained with p16 (*a-d*), which is not observed in WSIs stained with CD3/CD8 (*e*, *f*). This occurs because the whole epithelium contains neoplastic cells, causing the p16 biomarker to express. On the other hand, WSIs *e* and *f* show the expression of brown dots mostly outside of the epithelium. This is caused by the expression of the CD3/CD8 stains in CD3+ and CD8+ T cells. WSIs *a-d* correspond to the same slide, scanned with different scanners. Color and blurriness differences are the most recognizable between these. WSIs were scanned at 40x resolution (1 pixel = 0.2276µm).

3.4. Image preprocessing for dataset preparation

The image preprocessing methods used for dataset preparation usually depend on the quality and structure of the raw data, as well as the problem that is being addressed. In the context of digital pathology, however, the raw data (i.e. WSIs) usually shares a set of common characteristics: WSIs are usually very large images, they have a lot of background and they have annotated regions which may be relevant for the problem being addressed. The limited number of available WSIs and/or the small size of annotated regions may also be an important factor to consider. This section describes the methods that were used to generate a clean dataset from the image set that was previously described. Specifically, the process is divided into four steps: (a) *splitting the images into tiles*, (b) *labelling of tiles according to their class* (i.e. epithelium or non-epithelium), (c) *removing background tiles* and (d) *applying data augmentation to solve class imbalance.* A diagram of the complete process is available in Appendix A.

Splitting images into tiles

The size of WSIs makes it infeasible to use them directly for Deep Learning, so the first step is to split them into smaller tiles that can be used to train the DL models. The size of the tiles depends on the objective of the project (e.g. tissue classification may require bigger tiles than cell classification). Fig 3.4 allows to visualize the idea behind tile splitting. OpenSlide is a common library used to handle and extract regions from WSIs, and it was used for this work.

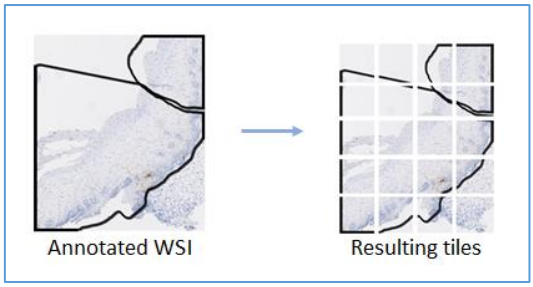


Fig. 3.4. Example diagram of WSI splitting into tiles.

Tile labelling according to their class

The second step is to label the tiles according to their respective class. In classification problems (i.e. problems where DL models are trained with labelled images), each tile needs to be labelled according to the annotations that were performed on the WSI. Fig. 3.5 allows to visualize that tiles inside the annotated region (the black contour line) are labelled as 1 while the remaining tiles are labelled as 0. Depending on the shape of the annotated region, some tiles may be only partially inside of it, so a criteria has to be specified for those cases. In this work, only tiles that were 100% inside the annotated region were labelled as 1 to make sure that DL models are trained with clean epithelium tiles, instead of tiles with partial non-epithelium or background.

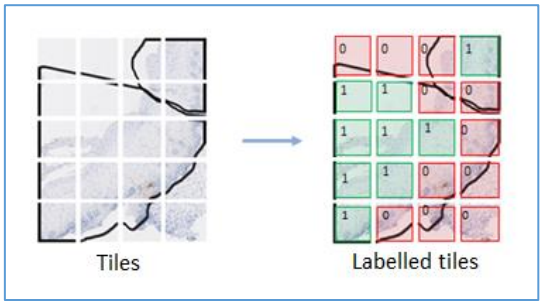


Fig. 3.5. Example diagram of the tile labelling result. In this case, only tiles that are 100% inside the annotated region are labelled as 1.

Removal of background tiles

The third step consists in the removal of tiles corresponding to background. Usually these tiles provide no relevant information for the DL models, and can therefore be removed using simpler image analysis techniques. The extraction of metrics from the color histogram of tiles can be an effective method to distinguish background from non-background tiles, as long as the background is homogeneous. Fig 3.6 shows two metrics that were used for this work: *standard deviation* and *proportion of pixels within a range from the mode*. The *standard deviation* describes the distribution of pixel colors and does not require further explanation. The *proportion of pixels within a range from the mode* was proposed ad-hoc for this work and describes the percentage of pixels that are within a specific range from the mode (i.e. the most frequent color). The size of the range has to be predefined in order to calculate the metric.

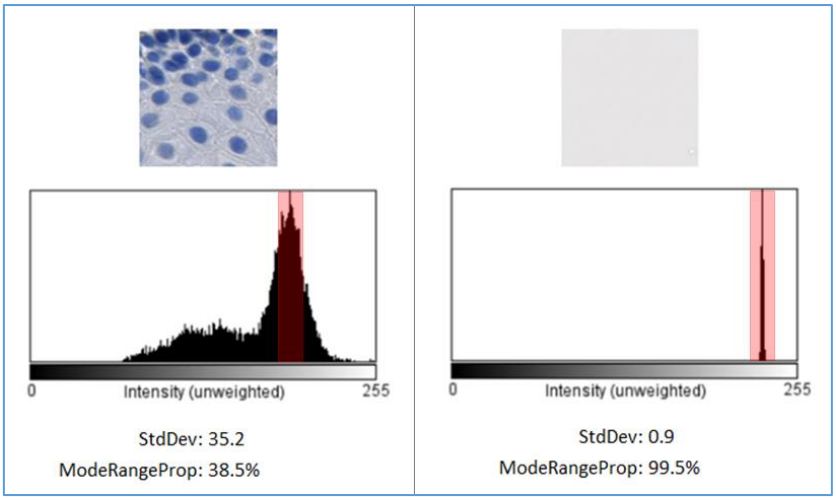


Fig 3.6. Example of a histogram analysis for two tiles. The *standard deviation* (StdDev) is calculated as the distribution of the pixel color in the image and the *proportion of pixels within a range from the mode* (ModeRangeProp) is calculated as the proportion of pixels within the red region over the total number of pixels in the image.

A final image analysis technique consists in the detection of edges within the image. The operation is a convolution between the image and a kernel that only highlights regions with high intensity gradient. The laplacian kernel is commonly used for this purpose. The result of the convolution can be observed in Fig. 3.7. Once the convolved image is obtained, the sharpest edge can be extracted as the pixel of highest value within the convolved image. This method can be used to recognize background with a color gradient, as described in the results section of this work.

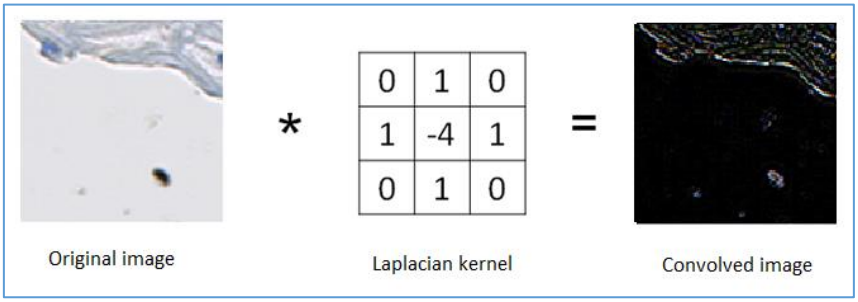


Fig. 3.7. Example of the convolution of an image with a laplacian kernel. The resulting convolved image highlights the edges of the original image.

Data augmentation to solve class imbalance

Finally, a technique called data augmentation can be applied as a fourth step if the dataset is too small, or if there is a class imbalance (i.e. one class has a significantly smaller number of images than the others). Data augmentation can be used to artificially increase the size of a dataset without the need of new data. In the context of this work, data augmentation consisted in expanding the number of images by applying rotations and reflections of the original images. As observed in Fig 3.8, by applying three rotations (90°, 180° and 270°) and flipping them, the number of images can be expanded 8-fold.

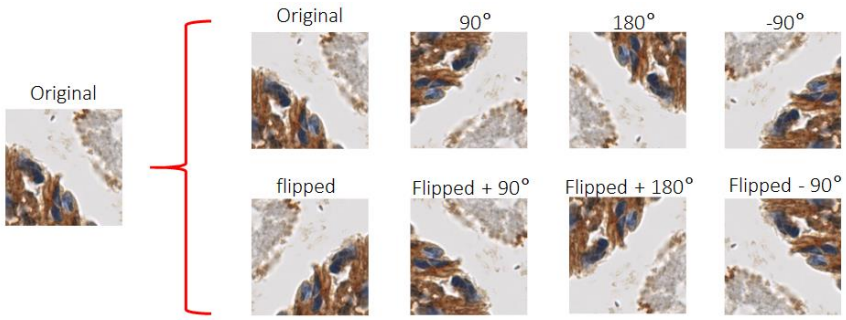


Fig. 3.8. Example of data augmentation for a tile. Eight copies of the original image with different rotations and flipping result from the process.

By applying the four steps described previously, an original set of WSIs can be processed into a dataset that can be used to train and test Deep Learning models.

3.5. Deep Learning

Deep Learning is a powerful machine learning tool that allows the processing of large amounts of data into models that are able to solve a multitude of complex problems. Through the process of training Deep Learning models, these are able to "learn" to recognize complex patterns.

This section describes technical basics of Deep Learning, starting with the concept of neural networks. Convolutional Neural Networks (CNNs) are then presented as an extension of neural networks which allows networks to be built deeper and obtain better performance without the

need for excessive computation power. A set of state-of-the-art CNNs are presented as well, which were used in the course of this work.

3.5.1. Neural Networks

The basic structure of a neural network is called perceptron (Fig. 3.9), a simplified model of a neuron which is able to solve linear problems. This is a *supervised* learning method (i.e. classification or regression method) where the input object (a set of input values) has a label indicating the correct output. This label is used to train the neuron so its own output is as close as possible to the correct one.

The perceptron is composed of a node that receives multiple input values (e.g. x_1, x_2,…, x_n), multiplies them by their respective weights (every input value has an associated weight, w_1, w_2, …, w_n) and passes them through an activation function which results in an output value. During the training process, multiple input objects are passed through the network, and the weights are updated based on the difference between the output and the label. This difference is calculated by means of a *loss function* (what we want to minimize), and the way in which weights are updated is determined by the *optimizer* (how we want to minimize it).

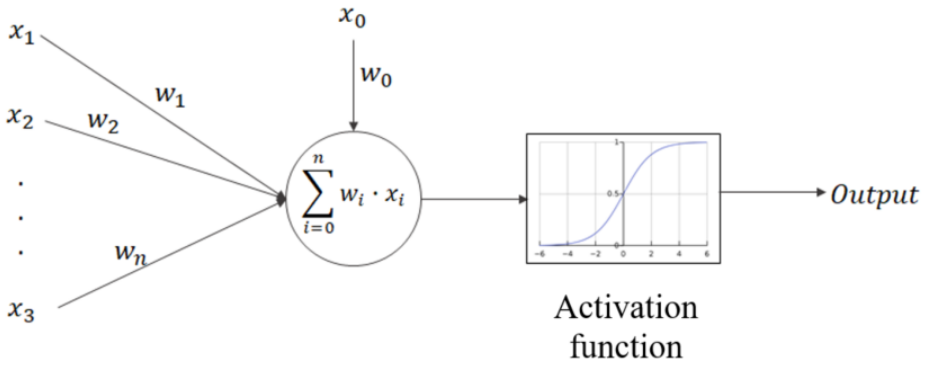


Fig. 3.9. Example of a perceptron. In the node, an aggregation function adds the weighted inputs, and this value is later passed to an activation function which generates an output based on the value of the weighted sum.

A Multilayer Perceptron (MLP) is a type of neural network composed of multiple layers of neurons that are connected to all neurons in the previous and posterior layers, except for the input and output layers, as described in Fig. 3.10. The hidden layers allow MLPs to solve more complex problems by applying non-linear combinations to the input.

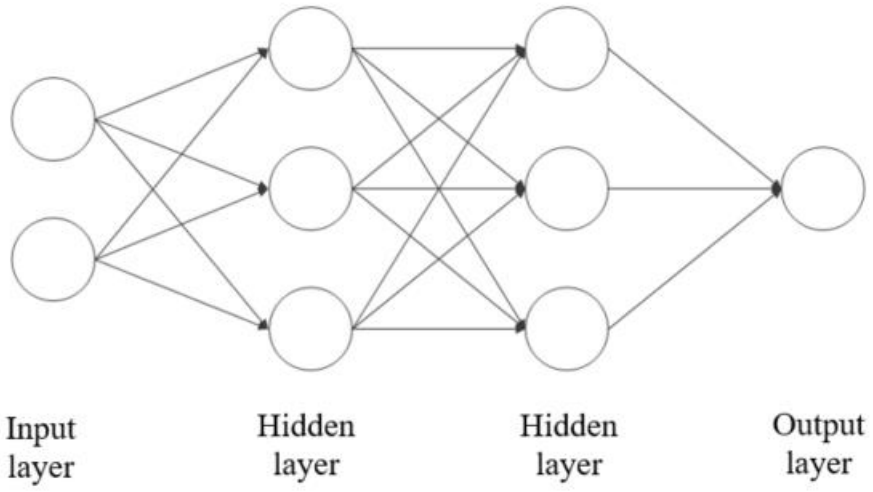


Fig. 3.10. Example of a MLP with two hidden layers. In this architecture, neurons (circles) from hidden layers receive the output of their previous layer's neurons, and their own output is fed into the next layer of neurons. This structure allows for non-lineal and mode complex pattern recognition processes.

When the network is trained, the weight of each connection between neurons is updated as it is presented with new input objects and their respective labels. This is done through an algorithm called backpropagation. The addition of new neurons and layers generates a significant increase in the number of weights to update, which is why it is not efficient to train big MLPs. In the context of image analysis, the number of input values associated with an image (i.e. the input object) is equivalent to the number of pixels and color channels, so for an image of 20x20px and 3 color channels (e.g. RGB), that would be 1.200 input values, making it unsuitable for these types of tasks. Just like in the case of single perceptrons, MLPs and neural networks in general use *loss functions* to calculate prediction error and *optimizers* to calculate the optimal weights for updating during training.

3.5.1.1. Loss functions

Loss functions are used to calculate the error in the model prediction, which corresponds to the difference between a model's output for the input data (e.g. an image) and the label for the same input data. It is expected for the error to decrease as the model is trained, meaning that the predicted output gets closer to the label. There are several different types of loss functions, and for this work a combination of two was used: Binary Cross-Entropy (BCE) and Kullback Leibler Divergence (KLD).

3.5.1.1.1. Binary Cross-Entropy (BCE)

Cross-Entropy has its origin in information theory, and it measures the number of bits (or the amount of information) that is needed to express an event according to the scheme used to describe this event. A scheme is designed with a probability distribution q and intends to describe events that occur with a real distribution p.

In the context of machine learning, p represents a categorical label of the input object (1 for the real image label and 0 for other possible labels), and q represents the likelihood that the model calculated for all possible labels given the input object. BCE is the same as Cross-Entropy, where an input object can only have one of two labels, and it is calculated as follows for a set of images:

$$BCE(p,q) = \frac{-1}{N}\sum_{i=1}^{N} p_i \cdot log_2(q_i) + (1-p_i) \cdot log_2(1-q_i) \quad (1)$$

$$p_i = real\ probability\ distribution\ for\ input\ object\ i$$
$$q_i = predicted\ distribution\ for\ input\ object\ i$$

Because p is always 1 or 0, the value of BCE is lowest as q is closest to p for image i. The reason is that the entropy of p is 0, and the lowest value BCE can have is the entropy of p. This is equivalent to say that BCE is lowest as the model prediction is closest to the real label.

3.5.1.1.2. Kullback Leibler divergence (KLD)

KLD is a similar measure of distance as BCE, but it measures the difference between the entropy for *p* and the entropy for *q* but it does not require p to be equal to 0 or 1. This value is the lowest as *p* and *q* are closest as well. It is calculated as follows:

$$KLD(p, p') = \frac{1}{N}\sum_{i=1}^{N} p_i \cdot log_2\left(\frac{p_i'}{p_i}\right) + (1 - p_i) \cdot log_2\left(\frac{1 - p_i'}{1 - p_i}\right) \quad (2)$$

$$p_i = distribution\ 1\ for\ input\ object\ i$$
$$p_i' = distribution\ 2\ for\ input\ object\ i$$

Because *p* does not have to be 1, it can be used in cases where the probability distribution of labels is not categorical.

3.5.1.2. Optimizers

Optimizers are the algorithms that handle how much a Deep Learning model is updated during training. In other words, they handle how the model "learns" to improve its prediction based from the error of their previous predictions[6]. During an iteration, optimizers take the error of the prediction (i.e. the difference calculated by use of the loss function, let's call it *J)* as input and calculate the modification that has to be applied to each weight in the model (i.e. the parameters that influence the model prediction) in order to minimize *J*. This section describes the optimizer used for this work, which is Stochastic Gradient Descent with Momentum (SGD+M) [61].

Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent is very commonly used in different Machine Learning algorithms. In this method, data instances (e.g. images or batches of images being fed to the model at a particular time) are sequentially presented to the model, and for every data instance *i*, the gradient of *J* with respect to the weights vector *W* is calculated. This calculated vector $\frac{\partial J}{\partial W}$ describes the rate of change that would have to be applied to the weights in order to perfectly predict the current data instance. However, applying $\frac{\partial J}{\partial W}$ directly to the weights is not convenient.

[6] "Introduction to Optimizers" from Algorithmia provides a friendly overall description of the optimization process in machine learning, as well as links to further resources. https://algorithmia.com/blog/introduction-to-optimizers

This particular calculation does not necessarily reflect the optimal weights for all data instances, but only for the current data instance. For this reason, a small step in the direction of the gradient is preferred, understanding that the collection of small steps for each data instance should make the model converge to the best possible weight values (i.e. to reach a point where it stops improving with new data instances). The learning rate ρ is introduced as the hyper-parameter that controls the "size" of these steps, as described in equation 3. ρ takes a small value between 0 and 1, commonly in an order of magnitude between 10^{-3} and 10^{-5}.

$$W_{i+1} = W_i - \rho \frac{\partial J_i}{\partial W_i} \tag{3}$$

Each data instance that is presented to the model corresponds to an *iteration* where the model updates its weights, and the presentation of all data instances available in the dataset is called an *epoch.* So models are usually trained for many epochs and the process is monitored using a validation set that is different from the training set. The training process is stopped when the error in the validation set converges or when it begins to increase (which is commonly caused by over-fitting). In order to avoid overfitting, an element called *decay* can also be added. Adding *decay* to the optimizer in the learning process basically means progressively shrinking the size of the steps in every iteration. By making the steps smaller, one can expect to reduce the possibility of overfitting after a long training by making the steps smaller along the training process. The hyper-parameter *d* for decay controls the rate at which the size of the step shrinks. Its value depends on the problem being addressed, but it is common to see values in the order of 1e-6.

Many new optimization algorithms have emerged to try to speed up the convergence process of the original SGD algorithm. One example of such algorithms, and the one used for this work is *Stochastic Gradient Descent with Momentum*.

Adding the momentum component (SGD+M)

The momentum component was added to the traditional SGD as a way to improve the learning process of Deep Learning models. The idea of SGD+M is that not only the gradient of the current

instance is used but also the momentum of the previous instances is taken into consideration. With this method, the "direction" of the gradient in the previous instances is added to the direction of the gradient in the current instance, as illustrated in Fig. 3.11. In this figure "momentum step" is a weighted sum of the gradient steps from the previous instances, and the "actual step" is a composition of both the current "gradient step" and the "momentum step". This is applied iteratively for every instance, meaning that the "momentum step" for the next instance will include the "gradient step" of the current instance. In SGD+M, the new hyper-parameter *m* is added. This new hyper-parameter controls the weight of the previous gradients in the actual step calculation.

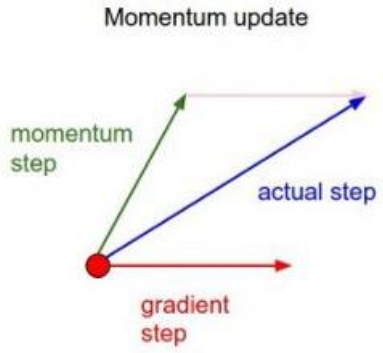


Fig. 3.11. Optimization "step" with SGD+M. In a traditional SGD, the learning "step" would correspond to the gradient step calculated for the current instance (red). With the addition of momentum, the "momentum step" (green) is added, so the actual step taken in this instance (blue) is a composition of both values.

3.5.2. Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are a type of Deep Neural Network commonly used for image analysis. They have been applied for image classification, segmentation and object recognition in video and images, among other uses. The most characteristic element of CNNs is its *convolutional layer*, which gives these types of networks their name. Convolutional layers extract features from the original image that become more complex further down the network (Fig. 3.12). The concept of Deep Learning emerges from the possibility of creating CNNs with many layers, which was not possible with previous neural networks such as the MLP.

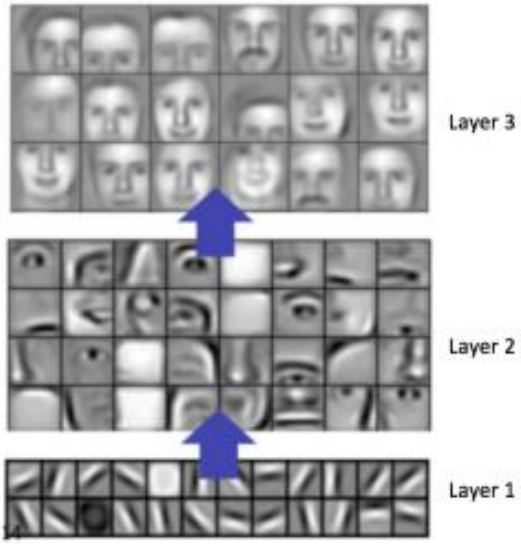


Fig. 3.12. Example of the increasing complexity of features on deeper layers. In this example, the first layer recognizes edges; the second layer starts recognizing more concrete elements such as eyes and noses; the third layer recognizes more complex objects such as whole faces. Source: [62]

Within a convolutional layer, a series of convolutions are applied to the layer input, as a form of abstraction from the original image to a feature space (e.g. edges, color distribution, size of elements, etc.). For each convolution there is a filter (or kernel) that generates a feature map, which is basically the result of the convolution between the input image and the filter. During the training process, filter values (i.e. weights) are updated as determined by the optimization algorithm, and based on the error calculated with the loss function. These layers, however, have a fewer number of parameters to train than fully connected layers. The only weights to update are the filter values, so this method is much more efficient than MLPs in image analysis [38].

It has been observed that increasing the network depth (i.e. number of layers in the network) improves model performance [63], [64]. The trade-off is the computational power necessary to train such models. Fig. 3.13 illustrates an example CNN, which receives an image as input, processes the information contained in the image (i.e. the pixel values from the image) and outputs it prediction.

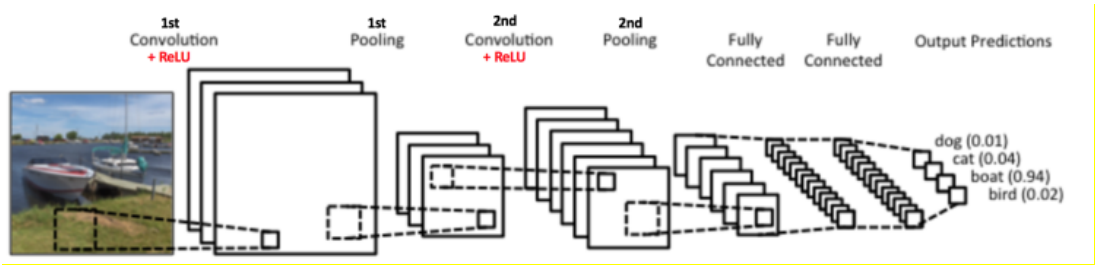


Fig. 3.13. Example of a Convolutional Neural Network. Features are extracted progressively on each layer, which are used by the network to predict an output for the image.

Another type of layer that is commonly used in CNNs is the *pooling layer*. Pooling is an operation which basically downscales the input from a previous layer by partitioning it into squares and outputting one value that summarizes the squares' information (Fig. 3.14). This reduces the amount of computation required to train CNNs under the assumption that no relevant information is lost.

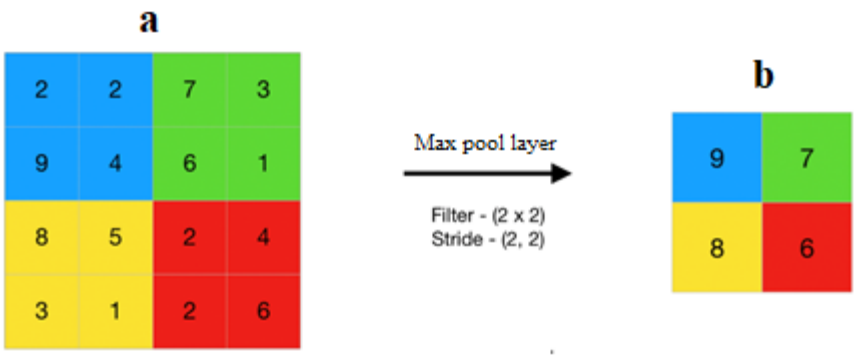


Fig. 3.14. Example of a Max-pooling layer. This layer downscales an input by partitioning it into squares and outputting the max value from each square. In this example, the colors of *a* represent the partitions, and the values of *b* correspond to the max value of each partition.

The example illustrated in Fig. 3.14 corresponds to a max-pooling layer, which outputs the maximum value for each square [65]. The filter size describes the size of the squares for pooling, and the stride describes the size of the step to extract the next step (a stride smaller than the filter would result in overlapping squares). Max-pooling is the most frequently used pooling layer in

Deep Learning. Average-pooling is another example of a pooling layer that can also be seen in Deep Learning applications [66].

3.5.3. Existing Deep Learning Networks

As described previously, there is evidence that in CNN performance increases with network depth[63], [64]. However, there is a trade-off between network depth and processing time. For this reason, new Deep Learning networks with innovative architectures have been developed to explore solutions for increasing depth without increasing processing time. Many of such networks exist in current literature, and three of them were used for this work: ResNet [67], InceptionV3 [64], [68], and InceptionResNetV2 [69].

3.5.3.1. ResNet

Sometimes when training a CNN, the optimal output for a layer may be the same as the layer's input (i.e. the output of the previous layer). However, updating the layer's weights to obtain such output involves a high processing time, since every weight must be calculated and recalculated on every iteration. In this context, residual networks (hence the name ResNet) are introduced as networks where a "shortcut connection" is added between layers as a new linear layer in parallel. Having this shortcut connection allows models to avoid the need to update weights on "unnecessary layers", by using the shortcut to directly output the output of the previous layer. ResNets do not have shortcut connections between each layer, but are built with sequentially connected residual "building blocks", as the one observed in Fig. 3.15.

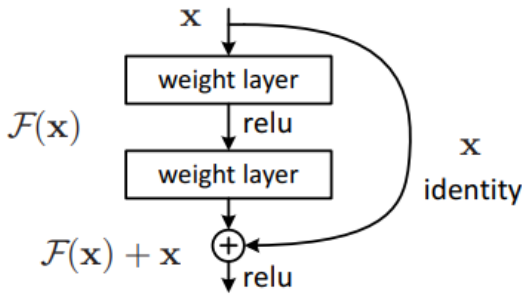


Fig. 3.15. Structure of the residual "building block" from the ResNet architecture. The shortcut connection (x identity) allows for the input X to be outputted by the building block. With this simple connection, the weight layers can be bypassed, saving the time that would be required to update their weights. Source: [70].

Each building block is composed by a set of layers and a shortcut connection that allows the residual block to output its input. This architecture proved to be quite effective in improving the predictive performance by allowing for the increase in network depth (i.e. more layers), without increasing the computational power and the processing time needed. For this work, the ResNet50 architecture was used, which is 50 layers deep.

3.5.3.2. InceptionV3

The Inception architecture was originally proposed by google. Its name actually derives from an internet meme that references the movie Inception[7]. Like ResNet, this architecture is built by sequentially connected blocks, as the one observed in Fig. 3.16. Inception offers two improvements on traditional CNNs simultaneously. First, it allows to apply convolutional layers with different filter sizes in parallel, allowing for a multi-level feature extraction on each block. Bigger filters (e.g. 5x5) extract more general features from the input, while smaller filters (e.g. 1x1) extract more local, position-dependent features. Second, it reduces the computational costs associated with weight updating by reducing the dimensionality. This dimensionality reduction is obtained by adding a 1x1 convolutional layer with fewer filters between the input and the convolutions used for feature extraction.

[7] The meme can be seen in this website, which was referenced in the original paper describing the Inception network. https://knowyourmeme.com/memes/we-need-to-go-deeper

With this architecture, it was possible to create deeper networks with high performance and a relatively low processing time.

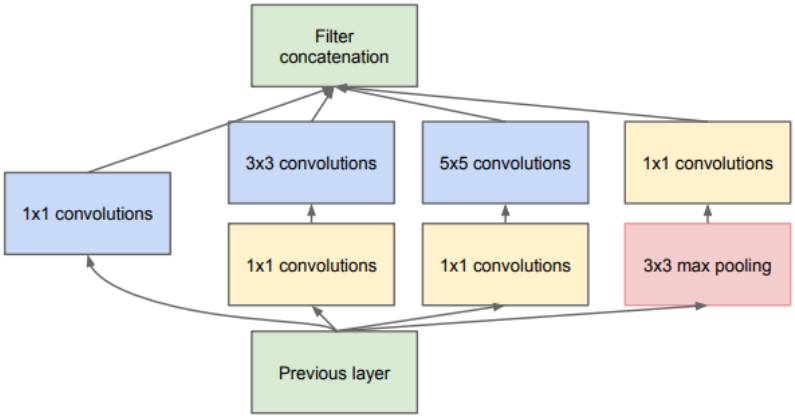


Fig. 3.16. Example of an inception block. The input from the previous layer is filtered by one 1x1, one 3x3 and one 5x5 convolutional layer (blue cells), as well as a max-pooling layer (red cells). The yellow cells correspond to the layers meant for dimensionality reduction. The outputs from these four paths are concatenated into one output to be received by the next inception block. Source: [64].

InceptionV3 is a variation of the original Inception architecture, developed by the same authors [68]. It follows the same principle of block-based multi-level feature extraction with dimensionality reduction. The difference is that this version contains blocks with different compositions of convolutional layers, which generate an even bigger dimensionality reduction and high predictive performance. In its tensorflow.keras implementation (the one used for this work), the architecture is 48 layers deep.

3.5.3.3. InceptionResNetV2

The InceptionResNet architecture, as its name would suggest, combines the principles of the Inception and the ResNet architectures. It was developed by the same team that developed the Inception architecture, and like Inception and ResNet, it consists of a set of sequentially connected blocks which are composed of a combination of convolutional layers. Within these

blocks, convolutional layers are connected in the same way as in the Inception architecture. Additionally, a shortcut connection is included to allow the bypassing of the convolutional layers. In its tensorflow.keras implementation (the one used in this work), the InceptionResNet architecture is 164 layers deep, which makes it significantly deeper than its predecessor architecture. Fig. 3.17 shows an example of the Inception-ResNet block.

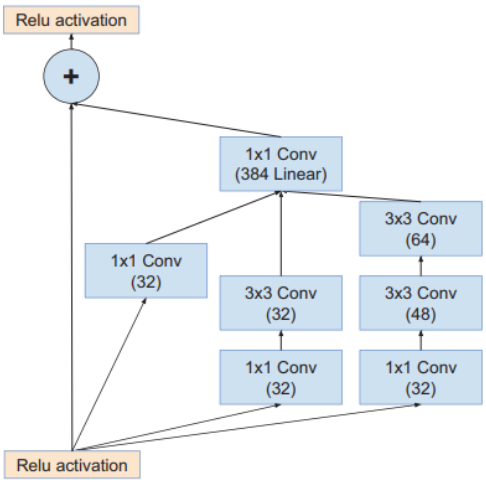


Fig. 3.17. Example of an Inception-ResNet block. The parallel convolutional layers for multi-level feature extraction are inherited from the Inception architecture. The shortcut connection that allows the block to be bypassed is inherited from the ResNet architecture. Source: [69]

3.6. Stability Training for robustness

In Deep Learning, robustness is a measure of how effective a model is when handling distorted inputs or inputs from different sources than the ones used for training. This is particularly relevant for image analysis, since it is common to find images with varying resolutions, noise or color variations. Even visually indistinguishable images can confuse CNNs and lead to wrong predictions [49] (Fig. 3.18). In order for models to be useful for real world applications, these should be as tolerant as possible to such distortions. Zheng et al. [49] introduces Stability Training as a method to improve model robustness during training.

Fig. 3.18. Effect of small image variations on CNN predictions. Visually indistinguishable differences can lead to wrong CNN predictions. In this case, two similar frames of a video lead to a significant difference in the CNN prediction of the class "fox" (0.27 v/s 0.63, respectively). Source: [49]

As described in this publication, the implementation of Stability Training is relatively straightforward for any base CNN (in this case, the authors use InceptionV3). A distortion layer is added to the network, which applies a predefined *distortion* on the original image, generating a distorted copy. Both the original and the distorted images are fed to the base CNN where a prediction is outputted for each. In the setup proposed by Laermann et al. [71] (Fig. 3.19), the loss function *L* is a composition of two loss functions (both described in section 3.5.1.1), as can be seen in the following equation:

$$L(p, q, q'|\alpha) = \sum_{i=1}^{n} BCE(p_i, q_i) + \alpha \cdot KLD(q_i, q_i') \tag{4}$$

$$p_i = real\ probability\ distribution\ for\ image\ i$$
$$q_i = predicted\ distribution\ for\ image\ i$$
$$q_i' = predicted\ distribution\ for\ distorted\ version\ of\ image\ i$$

The first loss function is calculated as the Binary Crossentropy (BCE) between the output for the original image and the ground truth (i.e. the label of the image). The second corresponds to the *stability component* and is calculated as the Kullback-Leibler Divergence (KLD) between the output for the original image and the output for the distorted image. The resulting loss *L* is the addition of both values, where the KLD is multiplied by a predefined parameter α which controls the weight of the *stability component* in *L*. This means that the *distortion* from the

distortion layer and the parameter α will have a strong impact on the calculation of L and the overall learning process of the CNN.

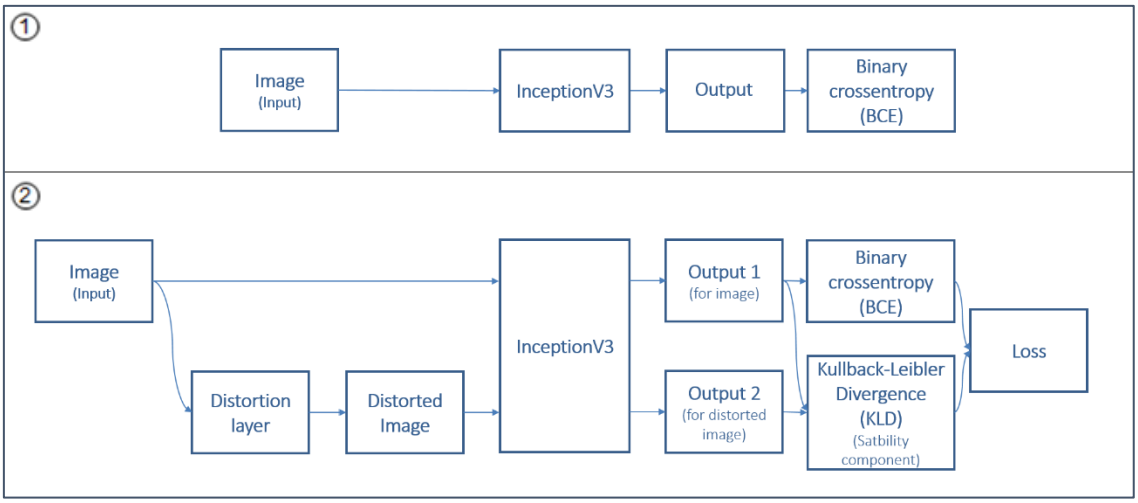


Fig. 3.19. (1) Schema for a regular CNN: Images are fed to the CNN (e.g. InceptionV3), and the output is used to calculate the loss (e.g. Binary Crossentropy). (2) Schema for a CNN for Stability Training: Images pass through a distortion layer which generates a distorted version of the original image. Both the original and the distorted images are fed to the CNN in parallel. BCE is used to calculate the loss for the original image, and KLD for the distorted image. An ensemble loss function (described in equation 4) is calculated with both values.

So far, this method has been used mainly to handle perturbations such as Gaussian noise (Fig. 3.20), information loss due to image compression, image downscaling and random cropping [49], [71]. However, in theory it could be used to handle any type of distortion or image perturbation as long as it can be emulated through the distortion layer. This method is described as an alternative to data augmentation and adversarial training, and has proven to be more efficient in terms of the computation power required.

Fig. 3.20. Example of a reference and distorted training image for stability training. Left: an original image. Right: a copy of the original image that was distorted in the distortion layer with Gaussian noise. Source: [49]

In the context of digital pathology, Stability Training seemed like a solid method to address image heterogeneity and train for robustness. Here, the main limitations are not necessarily compression loss or Gaussian noise, but are more related to variations in image properties such as color, blurriness, brightness and contrast. Luckily, these can be emulated without major complications on the distortion layer.

3.7. Area Under the ROC Curve (AUC) as a performance metric

In order to compare the performance between models, a performance metric is required. For the purpose of this work, the Area Under the ROC Curve (AUC) was selected. In order to understand the ROC curve and the AUC, first it is important to understand how a binary classifier system works. When a binary classifier system (such as a Deep Learning model designed to classify a region as "epithelium" or "non-epithelium") is used to predict the class from a specific input, the output is not one class or the other, but the likelihood that the input corresponds to the class of interest (e.g. the "epithelium"). This likelihood is a value between 0 and 1, so a value closer to one means that there is a high likelihood that the input corresponds to the class of interest and vice versa.

In order to make this binary classifier system useful, a decision has to be made regarding which is the appropriate likelihood threshold to discriminate between class of interest (positive) and

class of no interest (negative). For example, if a threshold is set to 0.1, then any input with a predicted likelihood of over 0.1 will be classified as positive. This may cause many negative inputs to be classified as positive which called false positives. On the other hand, a threshold set to 0.1 may cause the contrary effect: classifying many positive inputs as negative which are called false negatives. The True Positive Rate (TPR, also called "sensitivity") is a measure of the proportion of positives that corresponding to true positives, and is used to evaluate how "good" the classification within the positive inputs is. On the other hand, the False Positive Rate (FPR, also called "1-specificity") is the measure of the proportion of negatives that corresponds to false positives, and is used to evaluate how "bad" the classification within negative inputs is. Both calculations are illustrated in Fig. 3.21.

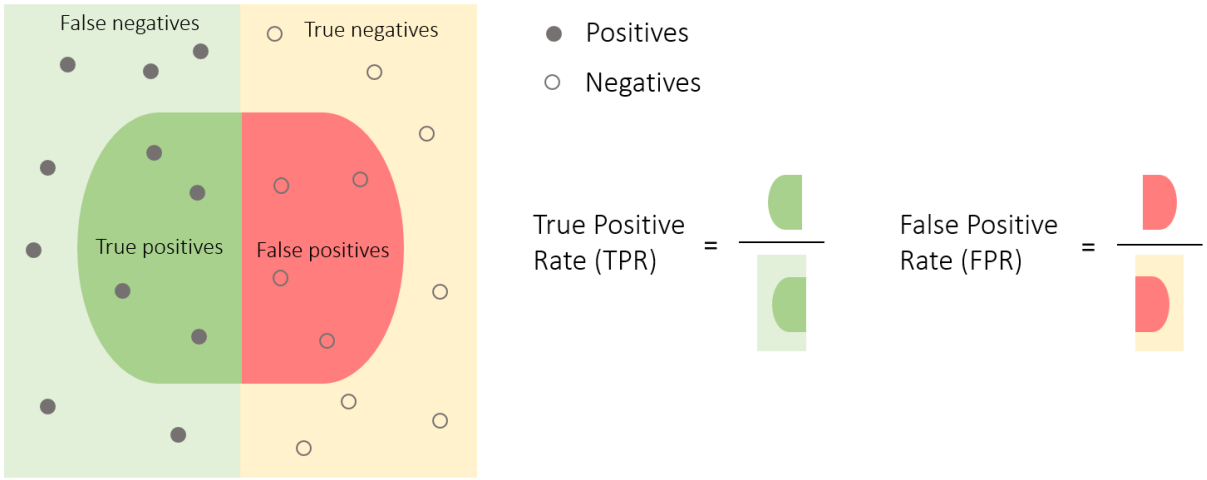


Fig. 3.21. Calculation of True Positive Rate (TPR) and False Positive Rate (FPR). TPR describes the number of predicted positives over the total number of positives, where 1 is a perfect TPR (i.e. all positives are predicted as such). FPR describes the number of false positives over the total number of negatives, where 0 is a perfect FPR (i.e. all negatives are predicted as such). Both metrics are commonly used to evaluate performance machine learning models.

The ROC curve is a graphical representation of the trade-off between true/false positives and true/false negatives for the different possible values of the likelihood threshold. It is created by plotting TPR and the FPR for a set of thresholds from 0 to 1, as can be observed in Fig. 3.22. It

allows to visually interpret how the classifier behaves, even though the likelihood threshold cannot be directly observed graphically.

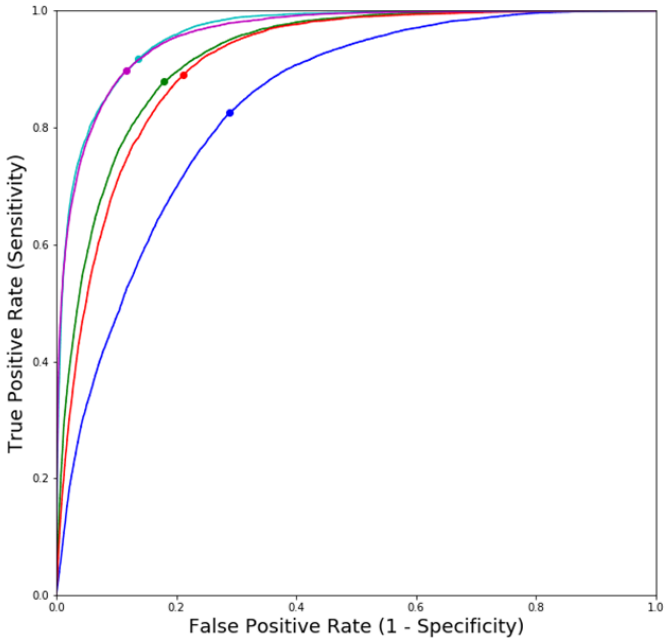


Fig. 3.22. Example of five ROC curves plotted in the same graph. This curve describes the TPR and FPR of 5 models for different likelihood cutoff thresholds. Since a model prediction is a likelihood between 0 and 1, predictions over the threshold are classified as positive and predictions under the threshold are classified as negative. A threshold closer to 0 causes more predictions to be classified as positive, even if these are negatives (i.e. high TPR but high FPR). On the other hand, a threshold closer to 1 causes more predictions to be classified as negative, even if these are positives (i.e. low FPR but low TPR).

The Area Under the ROC Curve (AUC), is a common metric used to evaluate classifier performance from the ROC curve, and is calculated exactly as its name suggests. The AUC can have a value between 0 and 1, since the graph is really a 1x1 square. The closer the AUC is to 1, the better, with 1 being a perfect segmentation performance (i.e. all positive inputs with a predicted likelihood of 1.0 and all negative inputs with a predicted likelihood of 0). The AUC is used to compare classifiers, and select the one with the highest AUC.

Even though the AUC is a useful metric to select the best performing machine learning model, the purpose of a binary classifier is, as its name suggests, assign a class to an image based on the model's predicted likelihood. In other words, a likelihood threshold has to be established to

decide whether the class corresponding to the image is 1 or 0. If the predicted likelihood of the image is under the threshold, the image corresponds to class 0, and if it is over the threshold then the image corresponds to class 1.

The decision of which threshold to use depends mostly on two factors: the *likelihood distribution* and the *purpose* of the binary classifier system. Intuitively, a threshold of 0.5 might seem reasonable; however, the likelihood distribution might not be centered at 0.5. This can be caused by imbalanced classes in the training set, or by an easier classification of one class over another.
On the other hand, the purpose of the classifier might be to prioritize finding all positives even if it means classifying some negatives as false positives. For these reasons, the selection of an optimal threshold is empirical and case-dependent.

There are multiple methods to select an optimal threshold based on the ROC Curve [72], [73]. Regarding the masks presented in this section, these were constructed with the optimal likelihood threshold for each model. This threshold was calculated using the Youden Index *J* [74]:

$$J = sensitivity + specificity - 1 \quad (6)$$

The likelihood cutoff threshold what maximizes J for the test set was selected. Visually, this threshold can be observed in the ROC Curve as the point that is closest to the top-left corner of the map.

4. Results

Chapter 4 describes the experiments and results associated with each of the activities executed during the project. Section 4.1 describes the methods that were evaluated to process the image set into datasets that were usable to train Deep Learning models. Section 4.2 describes the process of selecting a base Deep Learning architecture (i.e. InceptionV3 v/s ResNet v/s InceptionResnetV2) and the optimal hyper-parameter configuration. Section 4.3 describes the core of this work: training Deep Learning models using the Stability Training method (as described in section 3.5) and evaluating their performance on out-of-focus slides, slides scanned with three different scanners, and slides stained with three different IHC stains.

4.1. Background removal and dataset preparation

The dataset preparation is the first activity in any project involving images and Deep Learning. The quality of the data used to train a DL model has an enormous impact on its performance. For this reason, it is important to make sure that the dataset is clean and correctly labelled. The process, as described in section 3.4, involved (a) *splitting the images into tiles*, (b) *labelling of tiles according to their class* (i.e. epithelium or non-epithelium), (c) *removing background tiles* and (d) *applying data augmentation to the smallest class.*

The experimental component of this activity consisted in an exploratory analysis to select the best method for step (c) *removing background tiles*. Based on this analysis, the image set was processed into datasets for the training and testing of DL models. These datasets are the result of the dataset preparation process, and are described in this section as well. The images were split into 128x128px tiles, or approximately 29x29µm tiles given the 40x resolution with which WSIs were scanned. This size was selected mainly for two reasons: it is big enough to contain multiple cells of the same class, and it is small enough to extract a large number of tiles for model training.

4.1.1. Exploratory analysis for background removal

A Histogram analysis was performed on a small subset of tiles which were selected to represent possible occurrences of background and tissue. At first glance, it seemed that background tiles consisted in a plain homogeneous color without any element inside. This could make them easily identifiable by using the *standard deviation* as a metric, as observed in Fig. 4.1. In this figure, the histogram for the background tile (3) shows that most pixels have practically the same intensity, as expected; on the other hand, tiles corresponding to tissue (1 and 2) have pixels of varying intensity. The *standard deviation* of the pixel intensity for background tiles was significantly lower than for tissue tiles. After inspecting a set of approximately 50 random tiles from different slides, a standard deviation threshold of 5 was established: tiles with $std < 5$ were tagged as background.

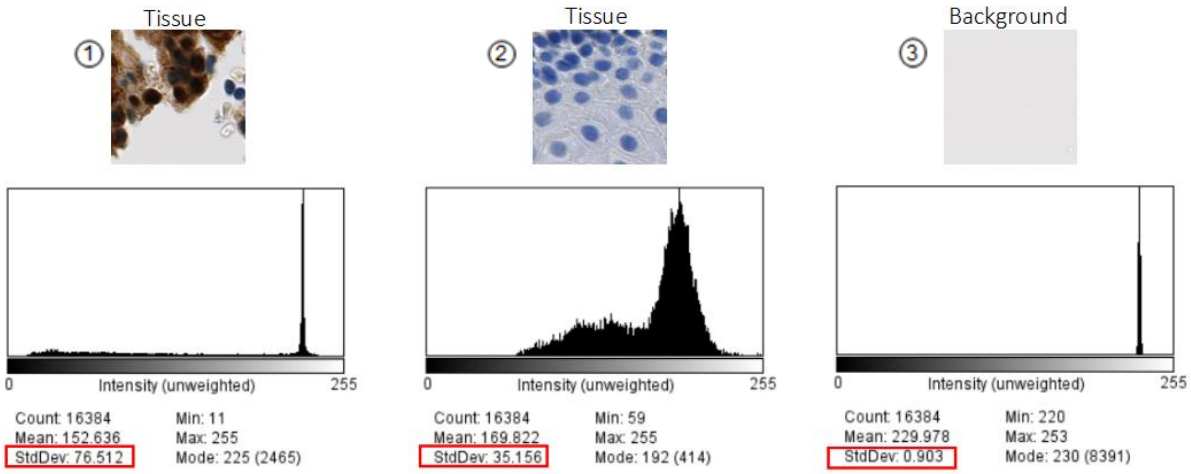


Fig. 4.1: Histogram and standard deviation of pixel intensity for tiles corresponding to tissue and background. It is observed that the standard deviation (marked in red) is higher for tissue tiles than for background tiles.

A closer examination of background tiles showed that the background is not always clean. Some tiles contained small dead cells, blurry smudges and other patterns that need to be recognized as background as well. Fig. 4.2 shows three examples of such tiles, and it can be observed that the *standard deviation* increases in all three cases, so it does not recognize these tiles as background.

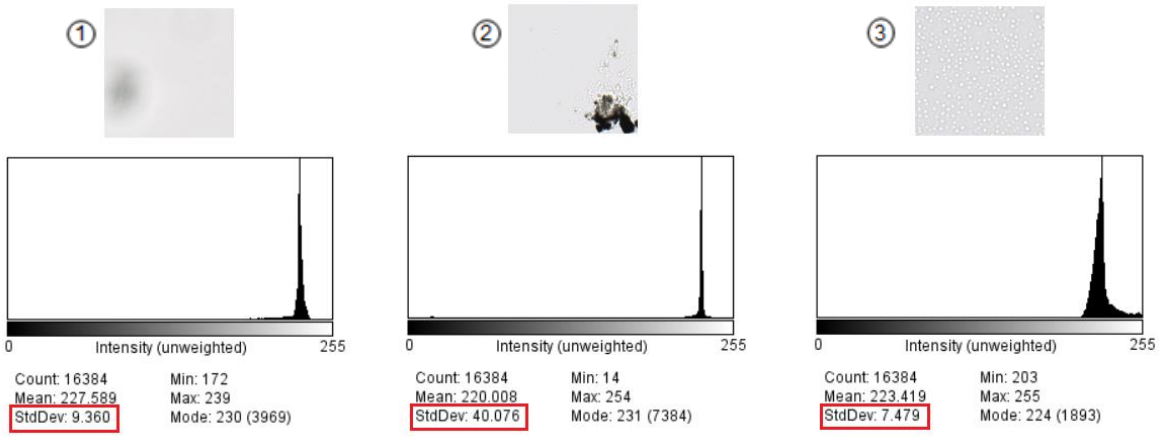


Fig. 4.2: Example of tiles corresponding to background which are not recognized by the filtering by standard deviation. The presence of particles (1, 2) and the inside of bubbles (3) can cause an increase in the standard deviation of their color histograms.

While observing the histogram for tiles with these patterns, it could be observed that even though the standard deviation was higher than expected, most pixels had an intensity that was close to the most frequent one (i.e. the mode). In other words, the majority of the pixels have a similar color. Based on this observation, the proportion of pixels within a specific range from the intensity mode was selected as a potential *second approach.* After analyzing a set of tiles with these characteristics, a range of ±10 from the mode and a proportion threshold of 0.85 were selected. This proportion was named Mode Range Proportion (MRP)[8], and is calculated as the number of pixels within the defined range over the total number of pixels in the tile. So, if a tile has a $MRP > 0.85$, it will be tagged as background. It can be observed in Fig. 4.3 that, for the three tiles presented previously, the MRP is higher than 0.85, so these would be correctly tagged as background.

[8] The concept of Mode Range Proportion was proposed in this work for convenience; it does not exist outside of this work.

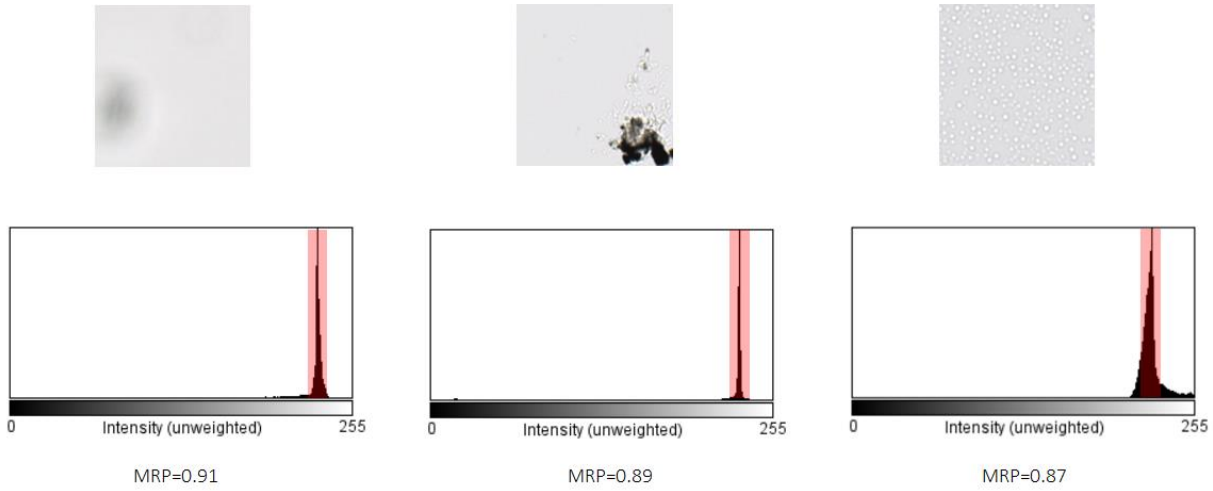


Fig. 4.3: Histogram of a background tile with a light gradient. The region in red depicts a range of ±10 intensity values from the mode. For these three tiles, the MRP is higher than 0.85.

On final examination of background tiles, a new pattern that had not been covered before was observed. Fig. 4.4 shows a tile that does not present a uniform background color. Instead, the background presents a color gradient, which is common in some WSIs. The proposed *second approach* does not recognize this tile as background, as the MRP is lower than 0.85. However, it can be observed that patterns such as the one presented in Fig. 4.4 do not show sharp edges, such as those that would be found in tiles with tissue, dead cells or patterns of this sort. For this reason, the absence of edges was added to the *second approach* as a condition to tag a tile as background.

In order to quantify the absence of edges, tiles are convolved by a laplacian filter, which is commonly used to identify edges in images. In the resulting convolved version of the original tile, the Sharpest Edge (SE) was defined as the pixel with the highest absolute value, and was selected as the criteria to tag tiles as background or non-background. After analyzing a set of tiles with this gradient pattern, a threshold of 40 was selected. This means that if the convolved version of a tile has a $SE < 40$, the tile was tagged as background. As it can be observed in Fig. 4.5, the background tile with a gradient has a SE lower than 40, while the non-background tile has a SE higher than 40.

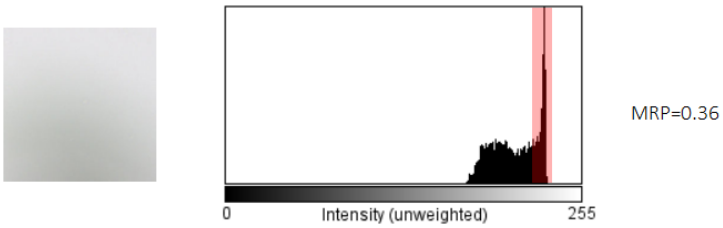


Fig. 4.4: Histogram of a background tile with a light gradient. The MRP is lower than 0.85, so this tile is not recognized as background using MRP only as the criteria.

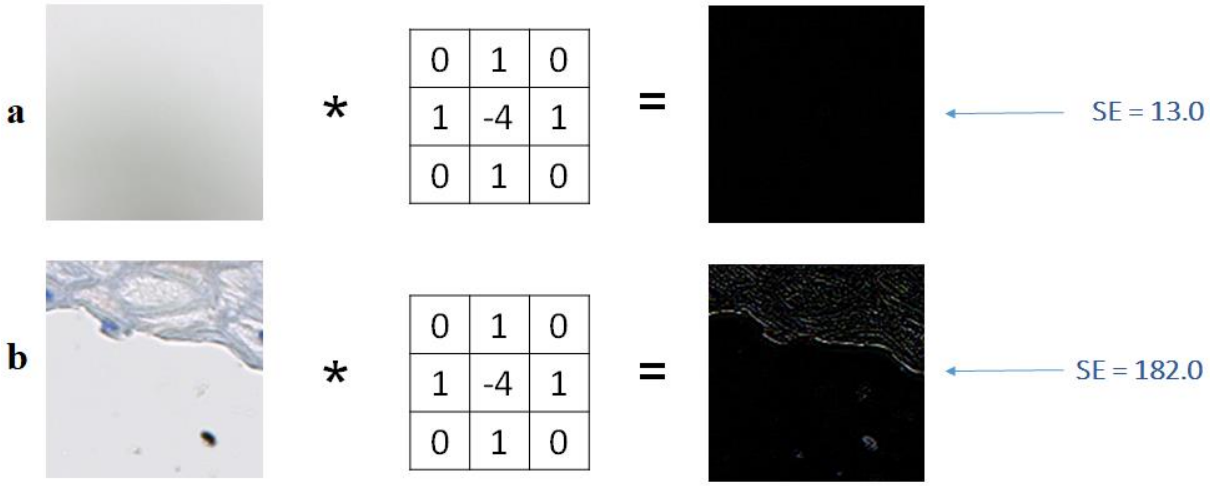


Fig. 4.5: Calculation of SE for (*a*) background tile with gradient and (*b*) non-background tile. The black images correspond to the convolved tiles. It can be observed that in the convolved version of tile *a* there are no edges, which is consistent with the low value of SE. On the other hand, the convolved version of tile *b* presents edges in the form of white and gray lines, which is why it has a high SE value.

This new condition proved to recognize background images with a color gradient, such as the one described in the above figure, so it was added to the *second approach*. Therefore, the final criteria used to tag tiles as background was:

$$(MRP > 0.85) \cup (SE < 40) \quad (5)$$

4.1.2. Resulting datasets for model training and evaluation

Once the criteria for background removal was selected, the images set described in section 3.3 was processed into datasets, following the method described in section 3.4. A total of 7 datasets were constructed, where one was used to train the Deep Learning models and the remaining 6 were used to test the models. Table 4.1 describes the characteristics of each dataset. It can be observed that the proportion between the number of epithelium and non-epithelium tiles is very different in the training set with respect to the testing sets. The reason is that data augmentation was applied only on the epithelium tiles of the training set.

Purpose	Stain-scanner abbrev.	Type of staining	Scanner	N° of WSIs	N° of epithelium tiles	N° of non-epithelium tissue tiles
Training	p16-HT	P16	NanoZoomer-HT	114	405.992	607.627
Testing	p16-HT	P16	NanoZoomer-HT	24	12.601	66.221
	p16-HT-out of focus	P16	NanoZoomer-HT (WSIs scanned intentionally out-of-focus)	24	9.798	54.170
	p16-XR	P16	NanoZoomer-XR	24	11.416	58.259
	p16-S360	P16	NanoZoomer S360	23	11.132	59.239
	CD3-HT	CD3	NanoZoomer-HT	24	11.297	69.095
	CD8-HT	CD8	NanoZoomer-HT	24	11.073	71.365

Table 4.1. Characterization of the datasets used for training and testing. One dataset was used for training, and six datasets were used for testing.

4.2. Base architecture selection and hyper-parameter fine-tuning

Multiple evaluations were carried out in order to select the best performing base architecture and to fine-tune a set of hyper-parameters. This activity served the purpose of ensuring an optimal Deep Learning model design to be applied for Stability training. The evaluations consisted in training Deep Learning models with different base architectures and hyper-parameter values, using a small subset of the HT-scanned-p16-stained dataset (Table 4.2).

N° of WSIs	15
N° of epithelium tiles	63.024
N° of non-epithelium tiles	87.707
Training/validation split	60/40

Table 4.2: Dataset subsample used for base architecture selection and hyper-parameter fine-tuning

The objective of this activity was to learn how the training process of the models changes when using different architectures and hyper-parameter values, as well as comparing them to select the best. For this purpose, using a small subset of the original training set seemed enough to make that analysis. Binary Cross-Entropy (BCE) was used as the loss function to train the Deep Learning models. The optimizer used was Stochastic Gradient Descent with Momentum (SGD+M). The metric for performance evaluation was the BCE of the validation set. In the case of the base architecture selection, training time was taken into consideration as well. All models were trained for 15 epochs using batches of 64 tiles.

The following subsections describe the base architectures and the hyper-parameter values that were evaluated, as well as the experimental results obtained.

4.2.1. Base architecture selection

There are many Deep Learning architectures available in the current literature. For the purpose of this work, three state-of-the-art architectures were evaluated: ResNet, InceptionV3 and InceptionResnetV2. A fourth, simpler CNN with five convolutional layers and three pooling layers (5-conv-3-pool) was also evaluated as a reference. Fig. 4.6 describes the validation BCE per epoch during the training process for each base architecture. It can be observed that, as expected, the BCE of the 5-conv-3-pool models is significantly lower than that of the state-of-the-art models, reflecting a worse performance over the epochs. ResNet50 shows the lowest and therefore best BCE, even though its results and its learning curve is similar to InceptionV3. Surprisingly, the BCE of InceptionResNetV2 was higher than that for InceptionV3 and ResNet50. Better results were expected as this architecture is deeper and contains a higher number layers than the others. However, it seems that InceptionResNetV2 was still improving at epoch 15, suggesting a potential improvement with more epochs. The same was observed in

model 5-conv-3-pool, so it could be argued that 15 epochs were not enough. However, its performance suggests that a significant improvement (i.e. closer to the other models) would take even longer.

Table 4.3 summarizes the validation BCE for the best epoch and the training time for the 15 epochs. It can be observed that InceptionResNetV2 required almost 8 hours for training, being the slowest of all architectures, which was expected given its higher number of layers. ResNet50 had the second highest training time and InceptionV3 the third, around 1 hour 10 minutes faster than ResNet50.

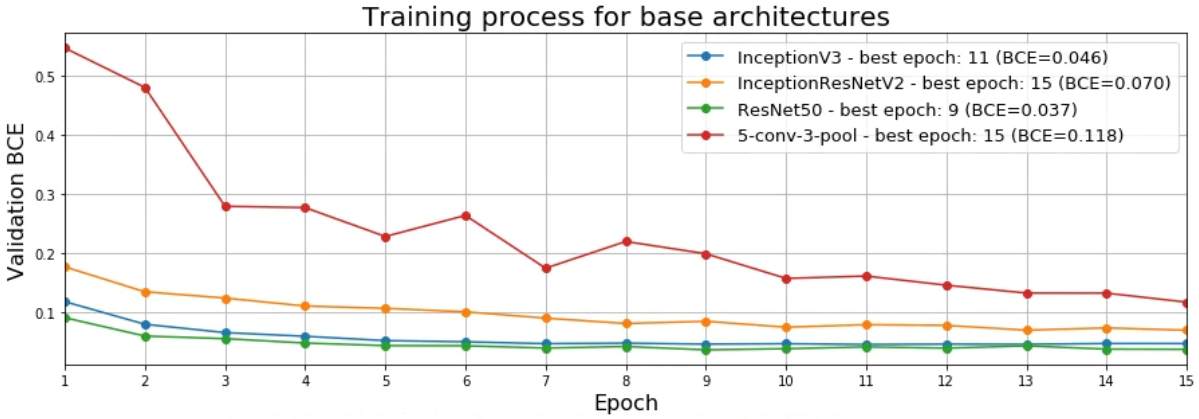


Fig. 4.6. The training process for each base architecture. 5-conv-3-pool shows a significantly higher (worse) validation BCE than the state-of-the-art. ResNet50 and InceptionV3 have a similar evolution of validation BCE through the epochs.

Base architecture	Validation Binary Crossentropy	Training time
ResNet50	0.037	05h 35m 30s
InceptionV3	0.046	04h 24m 08s
InceptionResNetV2	0.070	07h 58m 42s
5-conv-3-pool	0.118	03h 46h 28s

Table 4.3: Validation BCE of the best epoch per model, and training time for 4 architectures. ResNet50 had the best performance followed by InceptionV3, which also had a lower training time.

Even though ResNet50 had a slightly better validation BCE than InceptionV3, it was decided to continue working with InceptionV3. The difference in training time was an important variable to decide because of the time limitations of this work[9]. Additionally, research from TIGA in Deep Learning for digital pathology suggests that InceptionV3 performs well in tile classification problems [75]. This architecture was used for hyper-parameter fine-tuning, as well as for training the models with Stability Training (described in section 4.3).

4.2.2. Hyper-parameter fine-tuning

Hyper-parameter fine-tuning is a standard procedure in machine learning. The process consists in training and evaluating models with different hyper-parameter values in order to select the best-performing ones. Ideally all combinations of hyper-parameters would be evaluated; however, the number of combinations made the number of required models too high. For this reason, the selected approach consisted in establishing a base set of hyper-parameters and evaluating each hyper-parameter leaving all the rest fixed (*ceteris paribus*).

The selected hyper-parameters for fine-tuning were the optimizer hyper-parameters: learning rate (ρ), momentum (m) and decay (d). These were described in section 3.5.1.2. In this section, all models were trained using the InceptionV3 architecture and the default hyper-parameters ρ=1e-4, m=0.9 and d=1e-6, only changing the value of the parameter in study. Following the methodology of the previous section, the validation BCE was used as the performance metric for the training process, as shown in Fig. 4.7, 4.8 and 4.9.

[9] The HPC center from the University of Chile was available for multi-gpu processing; however, the learning process to use it and the modification required to the code made it faster to work with InceptionV3. However, the use of multi-gpu is strongly recommended for further work.

Learning rate (ρ)	Validation Binary Cross-Entropy
1e-5	0.097
1e-4 (base)	0.046
1e-3	0.039
1e-2	0.036

Table 4.4. Validation BCE of the best epoch per model. Results suggest a better (i.e. lower) BCE with a higher learning rate within a 15-epoch limit.

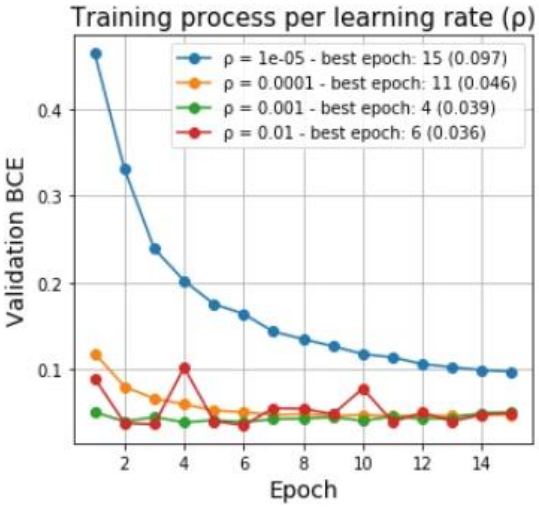


Fig. 4.7. Training process for the learning rate (ρ). Model *ρ=1e-5* takes a higher number of epochs to converge and does not converge by epoch 15. Model *ρ=0.1* shows an oscillating BCE, suggesting that ρ might be cause an unstable learning process.

Momentum (*m*)	Validation Binary Cross-Entropy
0.0	0.093
0.5	0.068
0.9 (base)	0.046

Table 4.5. Validation BCE of the best epoch per model. The best performing (i.e. lowest BCE) model was *m=0.9*

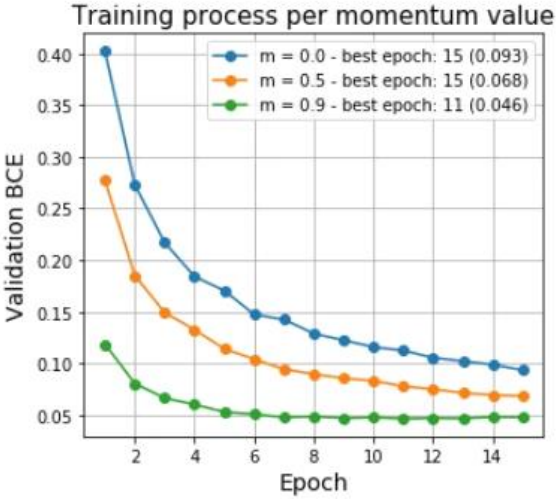


Fig. 4.8. Training process for momentum (*m*). Clearly the model with *m=0.9* has a faster learning process and reaches a lower BCE.

Decay (*d*)	Validation Binary Cross-Entropy
0	0.045
1e-6 (base)	0.046
1e-5	0.045
1e-4	0.048

Table 4.6. Validation BCE of the best epoch per model. No significant difference was detected on the different decay values.

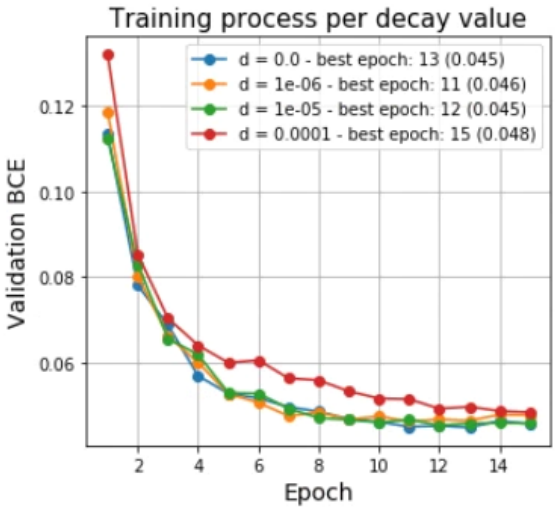


Fig. 4.9. Training process for decay (*d*). The learning process was almost identical for all models, although model *d=0.0001* showed consistently a higher (i.e. worse) BCE in the middle epochs.

It can be observed in Fig. 4.7 that the learning process for ρ=1e-5 is slow. Its best epoch is the last epoch (15), suggesting that it would require more epochs to converge. The remaining values for ρ converge significantly faster, especially for ρ=1e-3 and 1e-2. In the case of ρ=1e-2, some irregular spikes can be observed, suggesting a more unstable learning process caused by the bigger optimization "jumps" that a high ρ can cause. The model trained with ρ=1e-3 shows the second best results and the learning process seems more stable in terms of convergence to an optimal. For this reason, ρ=1e-3 was selected. Fig. 4.8 shows a significant difference in the learning process between the three values of momentum (*m*). Without a deeper analysis required, *m*=0.9 was selected. Regarding decay (*d*), no major difference could be observed in terms of the best validation BCE. A possible explanation is that models converge too fast for the decay parameter to have an effect in the learning process. In this case, the hyper-parameter value *d*=1e-6 was selected.

4.3. Stability training and evaluation of robustness

In order to evaluate robustness of the Deep Learning models, the Stability Training method was carried out as described in section 3.6. This evaluation consisted in training a set of Deep Learning models with different *distortion combinations* within the distortion layer, and with different values of the parameter α, which determines the weight of the *stability component* within the loss function. The purpose was to evaluate the effect of these parameters in the segmentation performance of the different test sets described in section 4.1.2. The models trained with Stability Training were compared against a model that was trained without Stability Training, in order to compare segmentation performance.

Distortion combination: Corresponds to the type and level of distortions applied in the distortion layer. The previous work in Stability Training focused on distortions such as information loss due to compression, Gaussian noise and scaling. This work, however, focuses on addressing differences that can be obtained from using different scanners and/or different IHC stains. As described in Table 4.7, the selected image properties addressed in this work were color, contrast, brightness and blur. For every image property, the distortion layer applies a random distortion within a range, so every distorted image has a different overall distortion than the rest even though the same distortion combination is being used.

	Distortion combination			
Image property	**1**	**2 (base)**	**3**	**4**
Color (red)	+ d ∈ [-5, 5]	+ d ∈ [-5, 5]	+ d ∈ [-20, 20]	+ d ∈ [-20, 20]
Color (green)	0	0	0	+ d ∈ [-20, 20]
Color (blue)	+ d ∈ [-5, 5]	+ d ∈ [-5, 5]	+ d ∈ [-20, 20]	+ d ∈ [-20, 20]
Contrast	· d ∈ [0.8, 1.2]	· d ∈ [0.8, 1.2]	· d ∈ [0.6, 1.6]	· d ∈ [0.6, 1.6]
Brightness	· d ∈ [0.7, 1.3]	· d ∈ [0.7, 1.3]	· d ∈ [0.5, 1.5]	· d ∈ [0.5, 1.5]
Blur (Gaussian kernel)	0	kernel size: 5x5 μ = 0.0 σ ∈ [0.0, 1.0]	kernel size: 5x5 μ = 0.0 σ ∈ [0.0, 5.0]	kernel size: 9x9 μ = 0.0 σ ∈ [0.0, 5.0]

Table 4.7. Distortion combinations that were evaluated. The cell color describes the level of distortion: green cells depict no distortion; yellow cells depict a low distortion; orange cells depict a medium distortion; red cells depict a high distortion. This distortion level comes from a visual appreciation of the effect it had on the original images.

Four distortion combinations were proposed, which progress in the intensity of the distortion, as can be observed in Fig. 4.10. In order to analyze the effect of the distortion combination, four Deep Learning models were trained, one for each combination. Each model was tested on the test sets described in section 4.1.2 in order to determine how the intensity of the distortion affects model performance for each one of the sets.

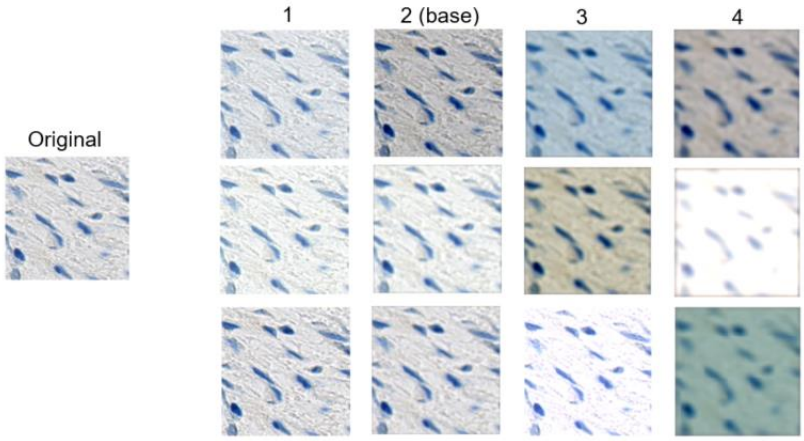


Fig. 4.10. Example set of distorted tiles corresponding to each distortion combination. Tiles from combination 1 are visually similar to the original, and the difference increases progressively on each combination. Finally, the tiles distorted with combination 4 seem visually as the most different from the original.

Weight of the stability component (α): The parameter α describes the weight of the stability component within the loss function of the Deep Learning models (described in equation 4, section 3.6). This parameter determines how strong the penalization associated with a poor prediction of the distorted image is. The papers used as reference for Stability training [49], [71] evaluated α values between 0.001 and 10. For this work the evaluated values were from 1 to 100, as well as α=0 (i.e. the model with no Stability Training) as reference.

The values for both distortion combination and α for each model are summarized in Table 4.8. All distortion combinations were evaluated by training these models with α=1 and all values for α were evaluated by training these models with distortion combination 2. This means that a total of 8 models were trained. Model 0 was trained without Stability Training.

Model	α	Distortion combination
0 (no Stability training)	0	0 (no distortion)
1	1	1
2 (base model)	1	2
3	1	3
4	1	4
5	2	2
6	10	2
7	100	2

Table 4.8. Evaluated values for the distortion combinations and parameter α for each model.

Fig. 4.11 describes the learning process of the Deep Learning models with Stability Training. Both an original image and a distorted version of the image are fed in parallel to the base architecture (in this example InceptionV3). An output is obtained for both the original and the distorted image, and the loss is calculated as described in section 3.6 (equation 4). This loss forces the Deep Learning model to reduce the difference between the output of the original image and the output of the distorted image, thus increasing the model's predictive robustness to distortions.

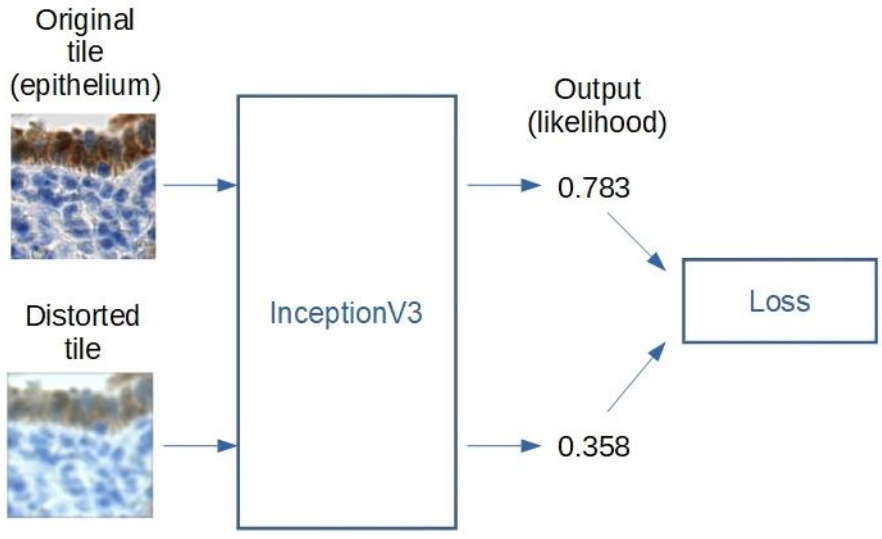


Fig. 4.11. Example of the Stability Training method on tiles extracted from WSIs. The original tile (in this example a tile corresponding to epithelium) and its distorted version are fed to the InceptionV3 network. The likelihood is outputted for each tile, which is used to calculate the loss as described in section 3.6.

All models were trained using the complete training set, built from 114 p16-stained WSIs scanned with the NanoZoomer-HT. The training set was split 60/40 for training/validation, and the validation set was used to select the best performing epoch for each model. This is the same method for epoch selection used for the base architecture selection and hyper-parameter fine-tuning.

4.3.1. Experiments

Once the Deep Learning models were trained using the different distortion combinations and values for α (including the model with α=0 and distortion combination 0, which is the model without Stability Training), these were tested on the testing datasets described in section 4.1.2. All models were trained using the same training set, which consisted in WSIs scanned in-focus with the NanoZoomer-HT, from slides stained with the p16 biomarker (Table 4.1).

The first pair of testing datasets corresponds to p16-stained WSIs scanned with the NanoZoomer-HT (i.e. the same type of WSI used for training). The second pair of testing datasets corresponds to p16-stained WSIs scanned with the NanoZoomer-XR and the NanoZoomer S360. The third and final pair of testing datasets corresponds to WSIs scanned with the NanoZoomer-HT and stained with CD3 and CD8.

E.1. p16-stained WSIs scanned with the NanoZoomer-HT: This experiment consisted in evaluating the trained models on two test sets: The set of *WSIs scanned in-focus* (E.1.1) and the set of *WSIs scanned out-of-focus* (E.1.2). The set of *WSIs scanned in-focus* is composed of the same type of WSIs that were used for training, allowing for a base performance to be established. The set of *WSIs scanned out-of-focus* was also included to evaluate how well models with Stability Training perform on WSIs with blurriness caused by a poor focus during the process of digitization. The relevance is that it is not uncommon to find blurry regions in WSIs.

E.2. p16-stained WSIs scanned with the NanoZoomer-XR and NanoZoomer S360: This experiment consisted in evaluating the trained models on datasets built from p16-stained WSIs that were scanned with the NanoZoomer-XR (E.2.1) and the NanoZoomer S360

(E.2.2). The models were not trained with WSIs from scanners other than the NanoZoomer-HT, so with this experiment, the models' robustness to distortions caused by different equipment is evaluated. The relevance is that in real world applications a Deep Learning model should be able to perform well regardless of the scanner used to create the WSI. A model that can only be applied on WSIs from one scanner is not scalable.

E.3. CD3/CD8-stained WSIs scanned with NanoZoomer-HT: This experiment consisted in evaluating the trained models on datasets built from CD3-stained (E.3.1) and CD8-stained (E.3.2) WSIs that were scanned with the NanoZoomer-HT. The models were trained only with p16-stained WSIs, so with this experiment, the models' robustness to distortions caused by different IHC stains is evaluated. The relevance is that for a use case such as epithelium segmentation, the models should be able to identify the epithelium regardless of the type of stain that was used. A correct segmentation of epithelium on WSIs with different stains will enable the generation of quantitative indicators that are relevant for each specific stain.

4.3.2. Results for p16-stained WSIs scanned with NanoZoomer-HT (E.1)

The following section describes the results obtained when testing the trained Deep Learning models on p16-stained WSIs that were scanned with the NanoZoomer-HT. The metric used for performance evaluation was the Area Under the ROC Curve (AUC). The graphs that compare performance for α include models 0 (no Stability Training), 2 (base model), 5, 6 and 7, while the graphs that compare performance for distortion combination include models 0, 1, 2, 3 and 4. The parameters for α and distortion combination in each model were described in Table 4.8. The optimal threshold was calculated using the Youden Index, as described in section 3.7.

E.1.1. WSIs scanned in-focus

From a tile-wise perspective (meaning that the AUC was calculated over the total number of tiles that compose the test set), the results obtained for the segmentation of WSIs scanned in-focus (i.e. same type of WSIs used for training) show that there is no significant difference in performance between the models trained with Stability Training and the model that was trained without Stability Training. All models present a segmentation performance of 0.98 or higher,

which is considered to be a significant performance. Fig 4.12 shows the ROC curves to allow the comparison between the segmentation performance of models trained with different values of α (left) and different distortion combinations (right).

Even though there is not a significant difference in performance, a small improvement can be observed when increasing the value of α. All models trained with Stability Training (α=1, 2, 10 and 100) had a better segmentation performance than the model with α=0 (i.e. no Stability Training). Regarding the distortion combination something similar can be observed, where the model with no Stability Training (i.e. Distortion comb=0) had a lower performance than the models with Stability Training, with the exception of the model with Distortion comb=100 which had the lowest AUC. However, the difference between models is almost negligible. This could be explained by that fact that Stability Training does not generate significant improvements if the test set has no distortions with respect to the training set.

The optimal likelihood threshold was calculated using the Youden Index as described in section 3.7. The threshold that maximizes the Youden Index for all tiles in this dataset was selected. With the exception of the model α=100 (0.601), the calculated threshold of all the other models was between 0.151 and 0.398.

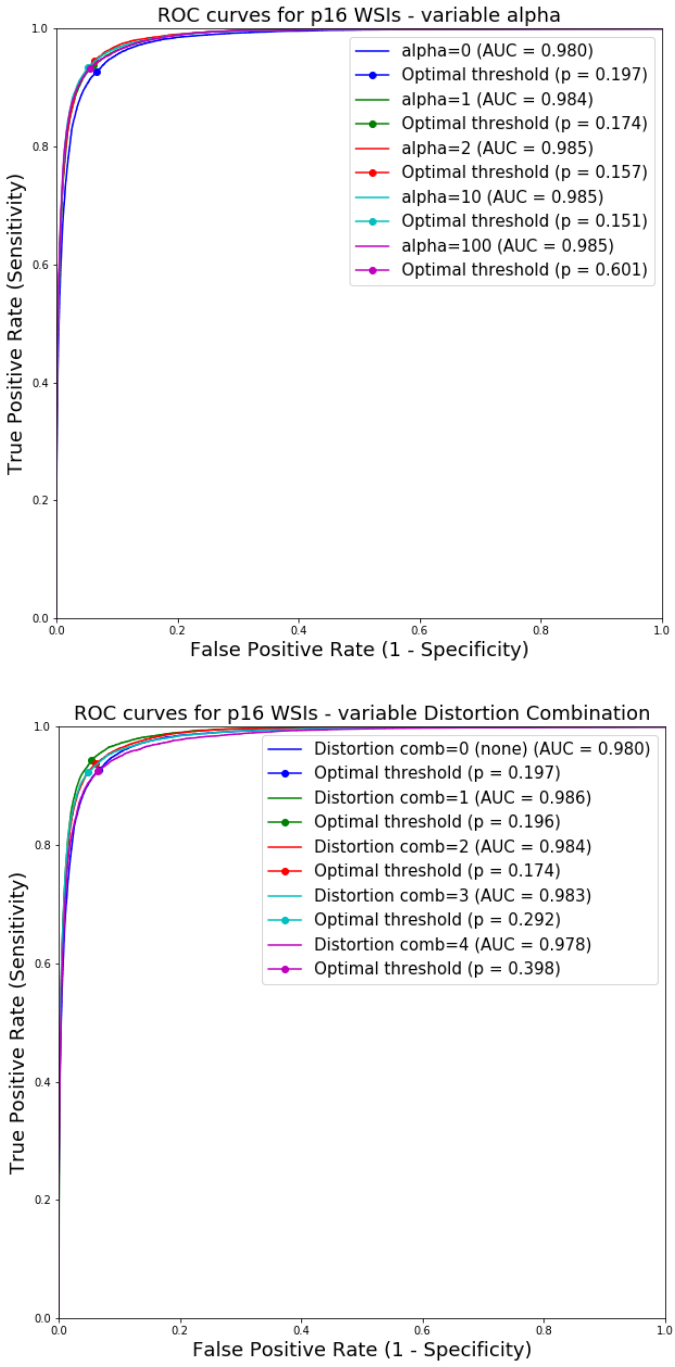
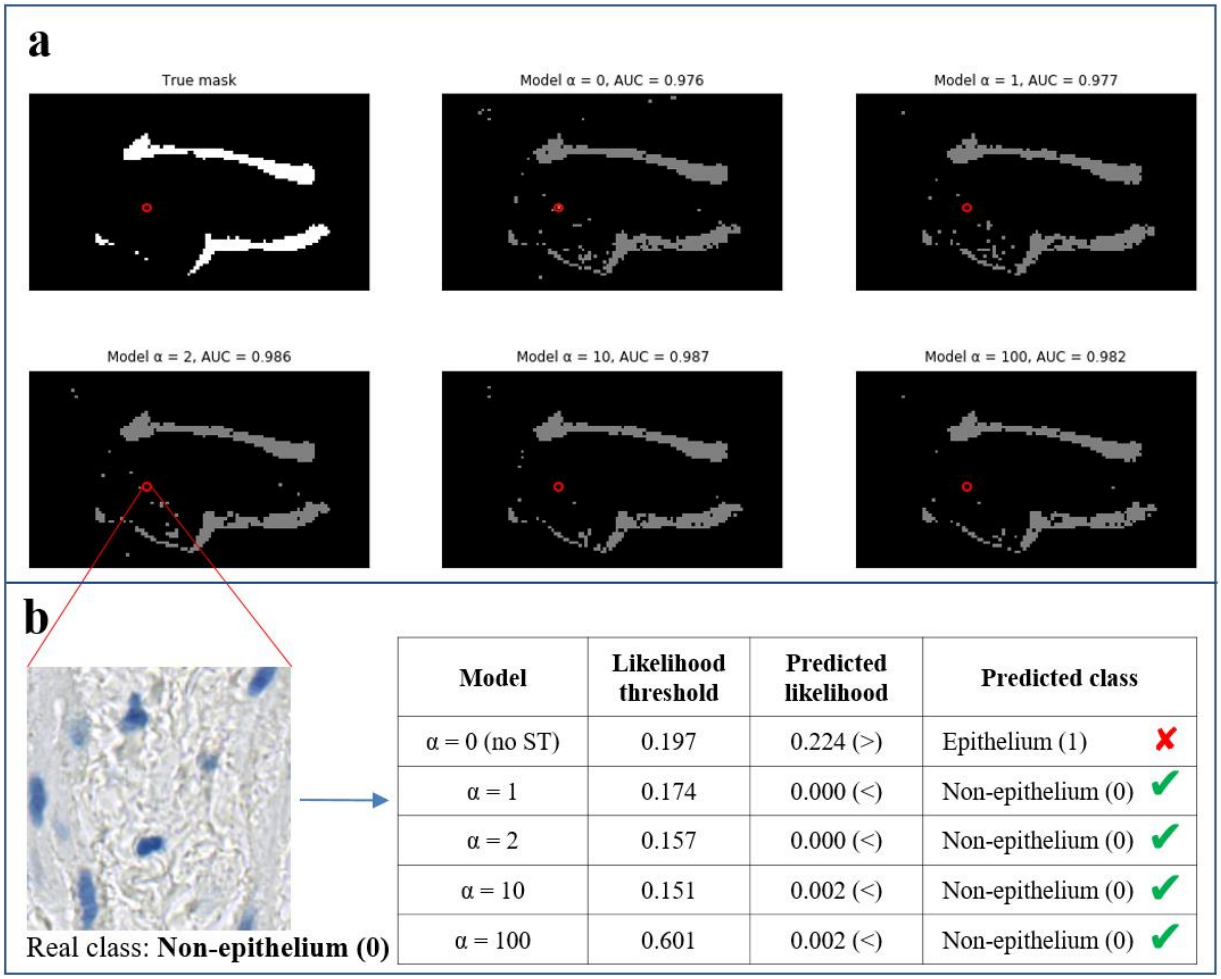


Fig. 4.12. (*top*) ROC curves to compare segmentation performance of models with different values for α on the p16-HT test set. No significant difference in performance can be observed between the different values of α. (*bottom*) ROC curves to compare segmentation performance of models with different distortion combinations. No significant differences in performance can be observed either.

Fig 4.13 (a) shows the segmentation of one of the WSIs from this test set for each value of α. A small improvement can be observed for models trained with Stability Straining (i.e. $\alpha \neq 0$). In this figure, the AUC corresponds to the performance for each tile that constitutes the image. Here, $\alpha = 10$ shows the best performance (AUC=0.987) and the improvement is most visible in the correct classification of tiles corresponding to non-epithelium, which is the black region in the "true mask" image. A boxplot of the AUC distribution per WSI can be observed in Appendix B, along with the mean and standard deviation of the AUC. An example tile is also presented in Fig. 4.13 (b), together with the predicted class from the model without Stability Training and the models with different values of α. It can be observed that the model without Stability Training incorrectly predicted the class "epithelium", while the remaining models correctly classified it as "non-epithelium".



Model	Likelihood threshold	Predicted likelihood	Predicted class
α = 0 (no ST)	0.197	0.224 (>)	Epithelium (1) ✘
α = 1	0.174	0.000 (<)	Non-epithelium (0) ✔
α = 2	0.157	0.000 (<)	Non-epithelium (0) ✔
α = 10	0.151	0.002 (<)	Non-epithelium (0) ✔
α = 100	0.601	0.002 (<)	Non-epithelium (0) ✔

Fig. 4.13. (a) Segmentations of a WSI from the p16-HT test set, by models with different values of α. The best performing model was α = 10 (AUC=0.987), while the worst performing model was α = 0 (no Stability Training) (AUC=0.976). These masks (and all masks presented in this section) were generated using the optimal cutoff threshold for each model. (b) An example tile shows a wrong classification from the model without Stability Training, while the models with Stability Training classify it correctly as "non-epithelium".

E.1.2. WSIs scanned out-of-focus (blurry)

From a tile-wise perspective, the results obtained for the segmentation of WSIs scanned out-of-focus show that the performance of the model trained without Stability Training is visibly lower than the performance of the models trained with Stability Training, as it can be observed in Fig. 4.14. Here, the model without Stability Training presented a significantly lower performance (AUC=0.876) when compared to the WSIs scanned in-focus (AUC=0.980). The models with

Stability Training also presented a lower performance here than with the WSIs scanned in-focus, however the difference is lower than in the model without Stability Training.

The better performance of models with Stability Training (as compared to the model without Stability Training) can be appreciated for all values of α, and for all distortion combinations. In all Stability Training models the AUC is over 0.9, with the exception of α=100. In the case of α, it can be seen that the AUC increases as α increases up until α=100. This suggests that there is a point where increasing the weight of the stability component becomes counterproductive for the model's learning process during training. It is interesting to observe that the shape of the curve for α=100 is less symmetrical than the rest, tending more towards a higher sensitivity than a higher specificity.

The optimal likelihood threshold was calculated using the Youden Index as described in section 3.7. The threshold that maximizes the Youden Index for all tiles in this dataset was selected. With the exception of the model with distortion combination 4 (0.562), the calculated threshold of all the other models was between 0.020 and 0.358.

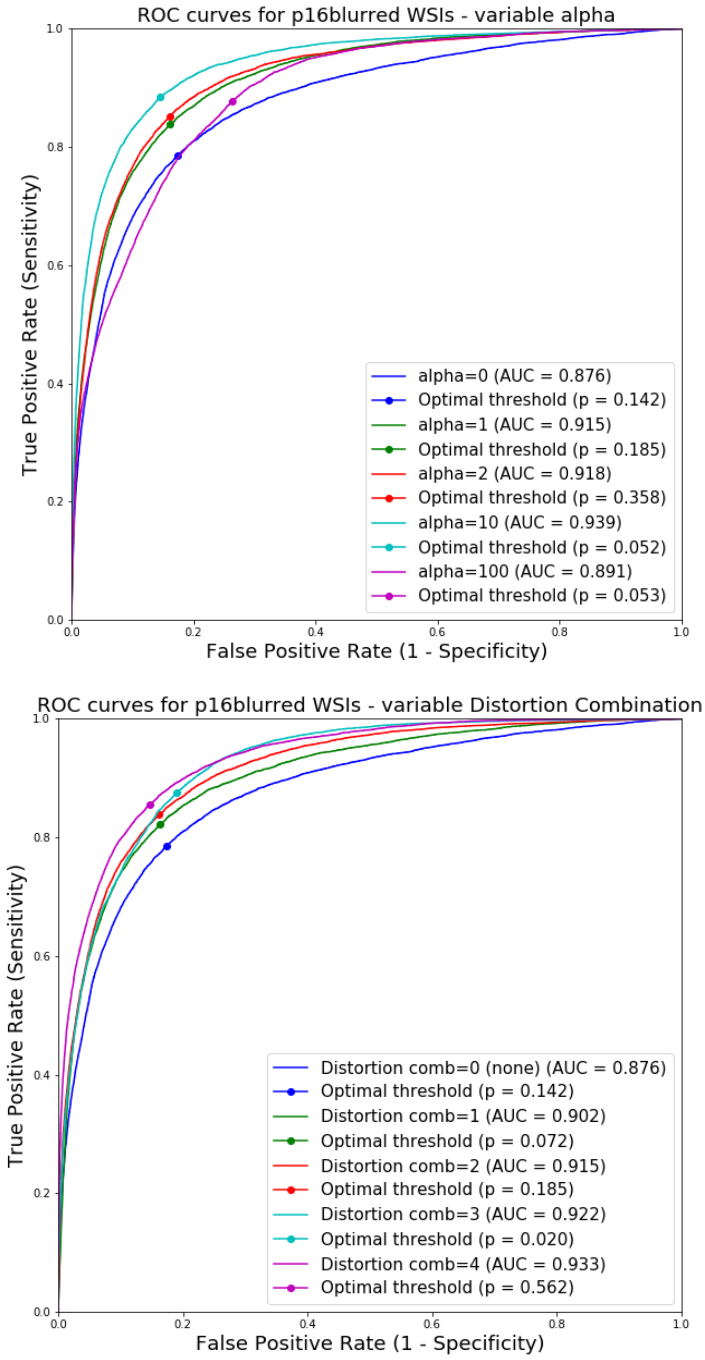
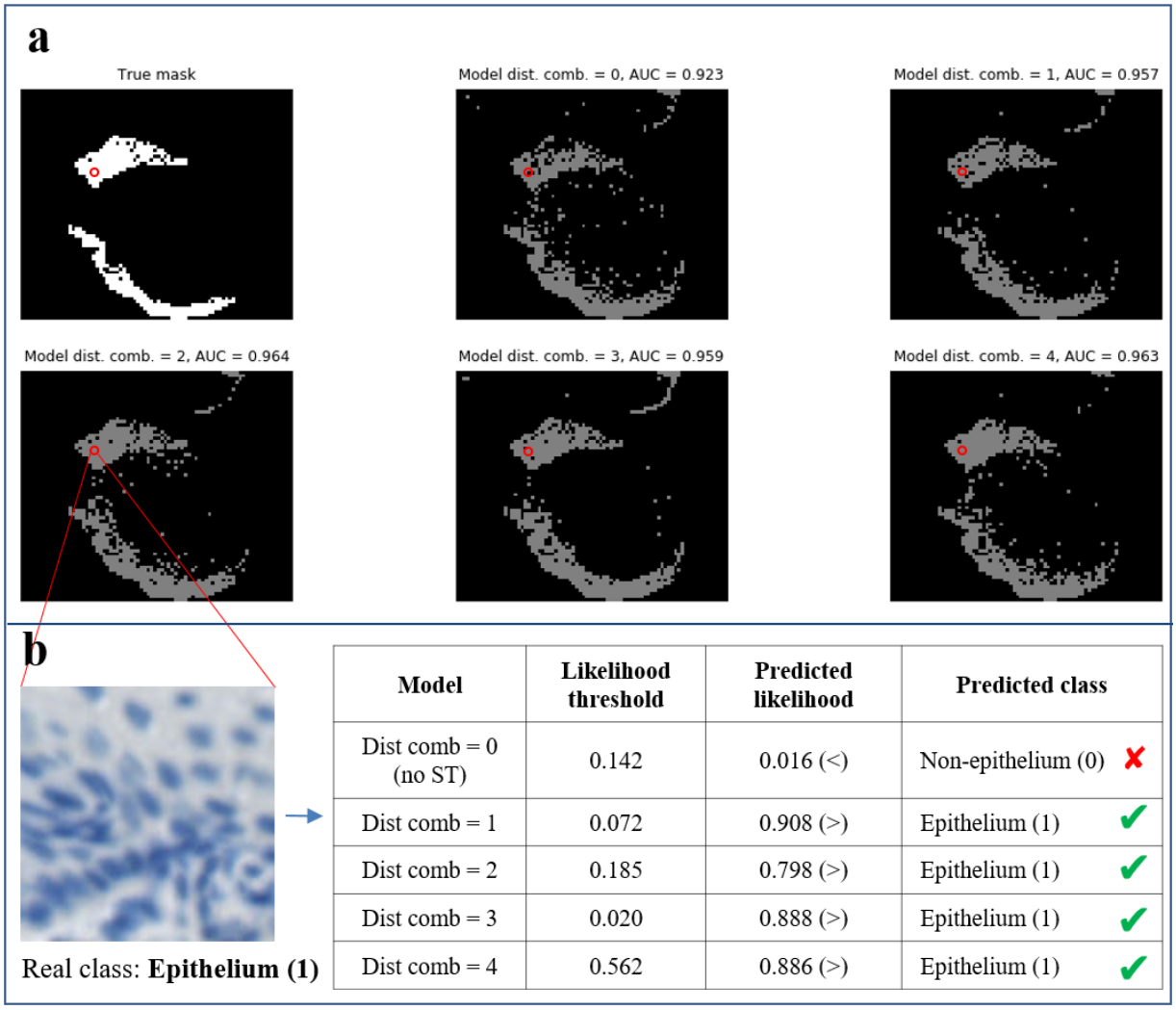


Fig. 4.14. (*top*) ROC curves to compare segmentation performance of models with different values for α on the p16-HT-out-of-focus test set. An improvement can be observed for models with Stability Training (α≠0). (*bottom*) ROC curves to compare segmentation performance of models with different distortion combinations on the p16-HT-out-of-focus test set. Increasing the level of distortion increases model segmentation performance.

In the case of the distortion combination, it can be observed that increasing the level of distortion (from 1 to 4) increases the model's segmentation performance. Contrary to previous suppositions, the best performing model corresponds to Distortion comb=4, with an AUC of 0.933. It was expected for a distortion combination such as 2 or 3 would have the highest AUC, since distortion combination 4 has a very high level of distortion; however, that was not the case.

It can be observed in Fig. 4.15 (a) that the models with distortion combination of 2 (AUC=0.964) and 4 (AUC=0.963) have the highest segmentation performances for this WSI. The lowest performance corresponds to the model without Stability Training, with an increase of non-epithelium tiles labelled as epithelium. These are represented by the small squares that should be black (as seen in the "true mask") but are gray instead. It can also be appreciated that all models recognize an object at the top right as epithelium. This corresponds to a small piece of fabric which was probably stuck during the preparation of the slide. Interestingly, the model with a distortion combination of 4 was able to correctly classify it as non-epithelium. A boxplot of the AUC distribution per WSI can be observed in Appendix B, along with the mean and standard deviation of the AUC. An example tile is also presented in Fig. 4.15 (b), together with the predicted class from the model without Stability Training and the models with different distortion combinations. It can be observed that the model without Stability Training incorrectly predicted the class "non-epithelium", while the remaining models correctly classified it as "epithelium".



Model	Likelihood threshold	Predicted likelihood	Predicted class
Dist comb = 0 (no ST)	0.142	0.016 (<)	Non-epithelium (0) ✘
Dist comb = 1	0.072	0.908 (>)	Epithelium (1) ✔
Dist comb = 2	0.185	0.798 (>)	Epithelium (1) ✔
Dist comb = 3	0.020	0.888 (>)	Epithelium (1) ✔
Dist comb = 4	0.562	0.886 (>)	Epithelium (1) ✔

Fig. 4.15. (a) Segmentations of a WSI from the p16-HT-out-of-focus test set, by models with different distortion combinations. The best performing models were distortion combination 2 (AUC=0.964) and 4 (AUC=0.963), while the worst performing model was distortion combination 0 (no Stability Training) (AUC=0.923). (b) An example tile shows a wrong classification from the model without Stability Training, while the models with Stability Training correctly classify it as "epithelium".

4.3.3. Results for p16-stained WSIs scanned with NanoZoomer-XR and S360 (E.2)

The following section describes the results obtained when testing the trained Deep Learning models on p16-stained WSIs that were scanned with the NanoZoomer-XR and the NanoZoomer S360. The metric used for performance evaluation was the AUC.

E.2.1. WSIs scanned with NanoZoomer-XR

From a tile-wise perspective, the results obtained on the p16-stained WSIs scanned with the NanoZoomer-XR show that Stability Training improves segmentation performance. This improvement can be observed in all models with the exception of α=100, as can be observed in Fig. 4.16. The performance of this model (AUC=0.884) was significantly lower than all others, suggesting once again that the high weight of the stability component might have affected the learning process of the model. The shape of the ROC curve also changed with regards to the other models, with a significant in False Positive Rate (FPR) starting from a True Positive Rate (TPR) of 0.5. This means that starting from that point, there is a high trade-off between TPR and FPR. Models with α=2 and α=10 show an almost identical ROC curve and had a higher performance than the remaining values of α.

Regarding the distortion combination, once again it can be observed that increasing the level of distortion improves the segmentation performance of the Deep Learning model. Stability Training with any of the distortion combinations had a better performance than no Stability Training, with distortion combinations 3 having the best performance (AUC=0.962) of all the evaluated models, including those trained for the analysis α.

The optimal likelihood threshold was calculated using the Youden Index as described in section 3.7. The threshold that maximizes the Youden Index for all tiles in this dataset was selected. With the exception of the model with distortion combination 4 (0.845), the calculated threshold of all the other models was between 0.044 and 0.390.

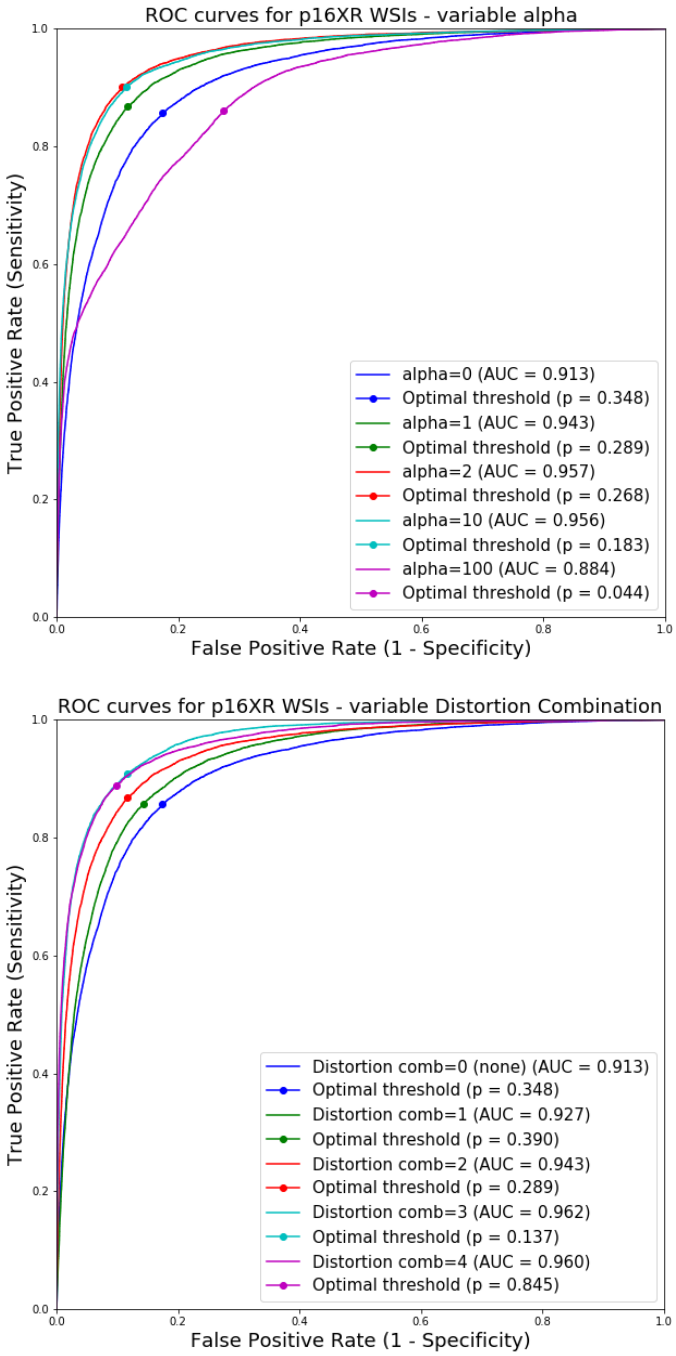
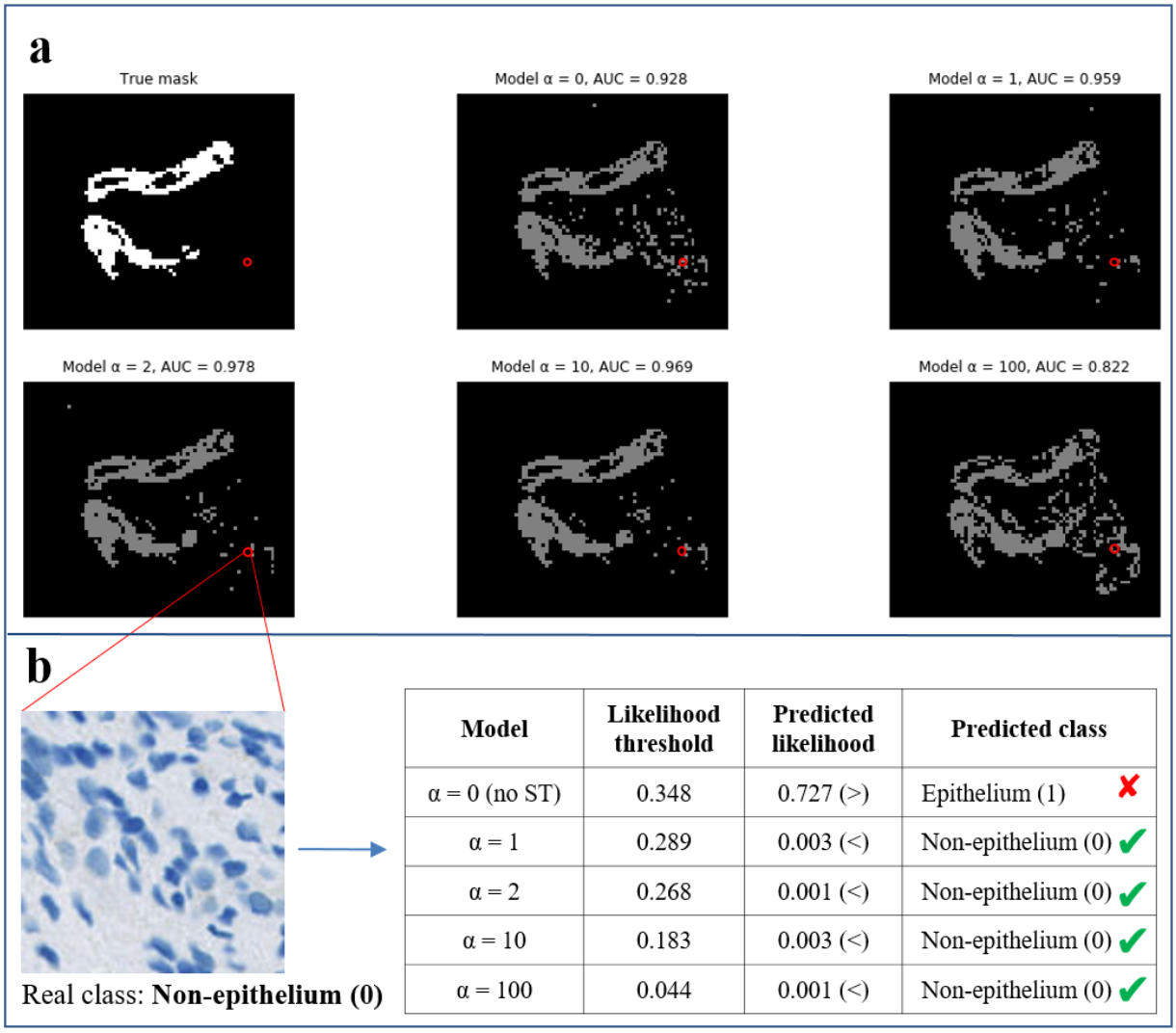


Fig. 4.16. (*top*) ROC curves to compare segmentation performance of models with different values for α on the p16-XR test set. Models with Stability Training have a higher performance with the exception of α=100. (*bottom*) ROC curves to compare segmentation performance of models with different distortion combinations on the p16-XR test set. Increasing the level of distortion increases model segmentation performance.

It can be observed in Fig. 4.17 (a) that the model with the best performance corresponds to α=2 (AUC=0.978). The main difference is the higher number of true negatives (i.e. correctly predicted non-epithelium tiles), which can be noticed by comparing the segmentation with the "true mask". The model with α=100 had the worst performance on this WSI (AUC=0.822), with a significant number of false negatives (i.e. wrongly predicted non-epithelium tiles). This is consistent with the ROC curve from Fig. 4.16, where the model with α=100 presented the worst overall performance on this dataset (AUC=0.884). A boxplot of the AUC distribution per WSI can be observed in Appendix B, along with the mean and standard deviation of the AUC. An example tile is also presented in Fig. 4.17 (b), together with the predicted class from the model without Stability Training and the models with different values of α. It can be observed that the model without Stability Training incorrectly predicted the class "epithelium", while the remaining models correctly classified it as "non-epithelium".

An example tile is also presented in Fig. 4.17, together with the predicted likelihood from two models: α=0 (no Stability Training) and α=2. It can be observed that the model without Stability Training incorrectly predicted the class "epithelium", while the model with α=2 correctly classified it as "non-epithelium". The model with distortion α=0 outputted a likelihood of 0.727 for the tile, which is higher than the model's optimal threshold of 0.348 (Fig. 4.16), therefore classifying the tile as "epithelium". On the other hand, the model with α=2 outputted a likelihood of 0.001, which is lower than its threshold of 0.268, therefore correctly classifying the tile as "non-epithelium".



Model	Likelihood threshold	Predicted likelihood	Predicted class
α = 0 (no ST)	0.348	0.727 (>)	Epithelium (1) ✘
α = 1	0.289	0.003 (<)	Non-epithelium (0) ✔
α = 2	0.268	0.001 (<)	Non-epithelium (0) ✔
α = 10	0.183	0.003 (<)	Non-epithelium (0) ✔
α = 100	0.044	0.001 (<)	Non-epithelium (0) ✔

Fig. 4.17. (a) Segmentations of a WSI from the p16-XR test set, by models with different values of α. The best performing models were α=2 (AUC=0.978), while the worst performing model was α=100 (AUC=0.822). (b) An example tile shows a wrong classification from the model without Stability Training, while the models with Stability Training correctly classify it as “non-epithelium”.

E.2.2. WSIs scanned with NanoZoomer S360

From a tile-wise perspective, the results for the WSIs scanned with the NanoZoomer S360 show a significant improvement in segmentation performance for models with Stability Training, as can be observed in Fig. 4.18. The results without Stability Training show a surprising poor performance (AUC=0.840), even though the differences between WSIs from the NanoZoomer-HT (i.e. the training set) and the NanoZoomer S360 are not so pronounced visually (mostly color differences can be observed). This further proves the importance of robustness in the application of Deep Learning in digital pathology. In the case of α, the best performing model corresponds to α=10 (AUC=0.952), significantly better than the performance with no Stability Training. For α=100, once again the performance had a strong decrease (AUC=0.900), with a spike in the FPR starting from the TPR of 0.6.

Regarding the distortion combination, the highest levels of distortion resulted in the best segmentation performances. The model trained with distortion combination 3 is overall the best performing model (AUC=0.962) followed closely by distortion combination 4 (AUC=0.960). It is interesting to see that clearly distortion combinations 3 and 4 performed similarly, as well as combinations 1 and 2. Both pairs of distortion combinations share the same color distortions and a similar level of blur (the blur from model 2 is very light, and visually almost negligible).

The optimal likelihood threshold was calculated using the Youden Index as described in section 3.7. The threshold that maximizes the Youden Index for all tiles in this dataset was selected. For this dataset, there was a significant variation in the threshold for each model. The model with distortion combination 4 had the highest threshold (0.911) followed by the models with α=2 and α=10 (0.661 and 0.612 respectively). The threshold of all the other models was between 0.037 and 0.375.

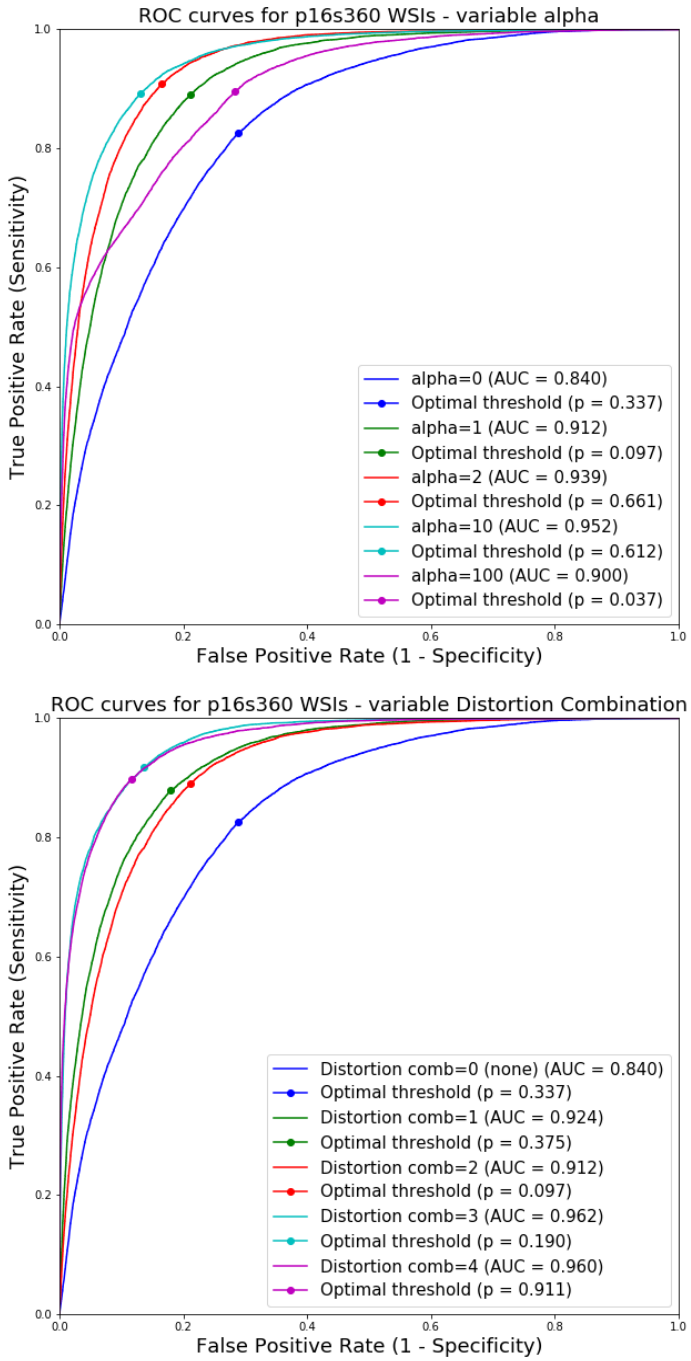
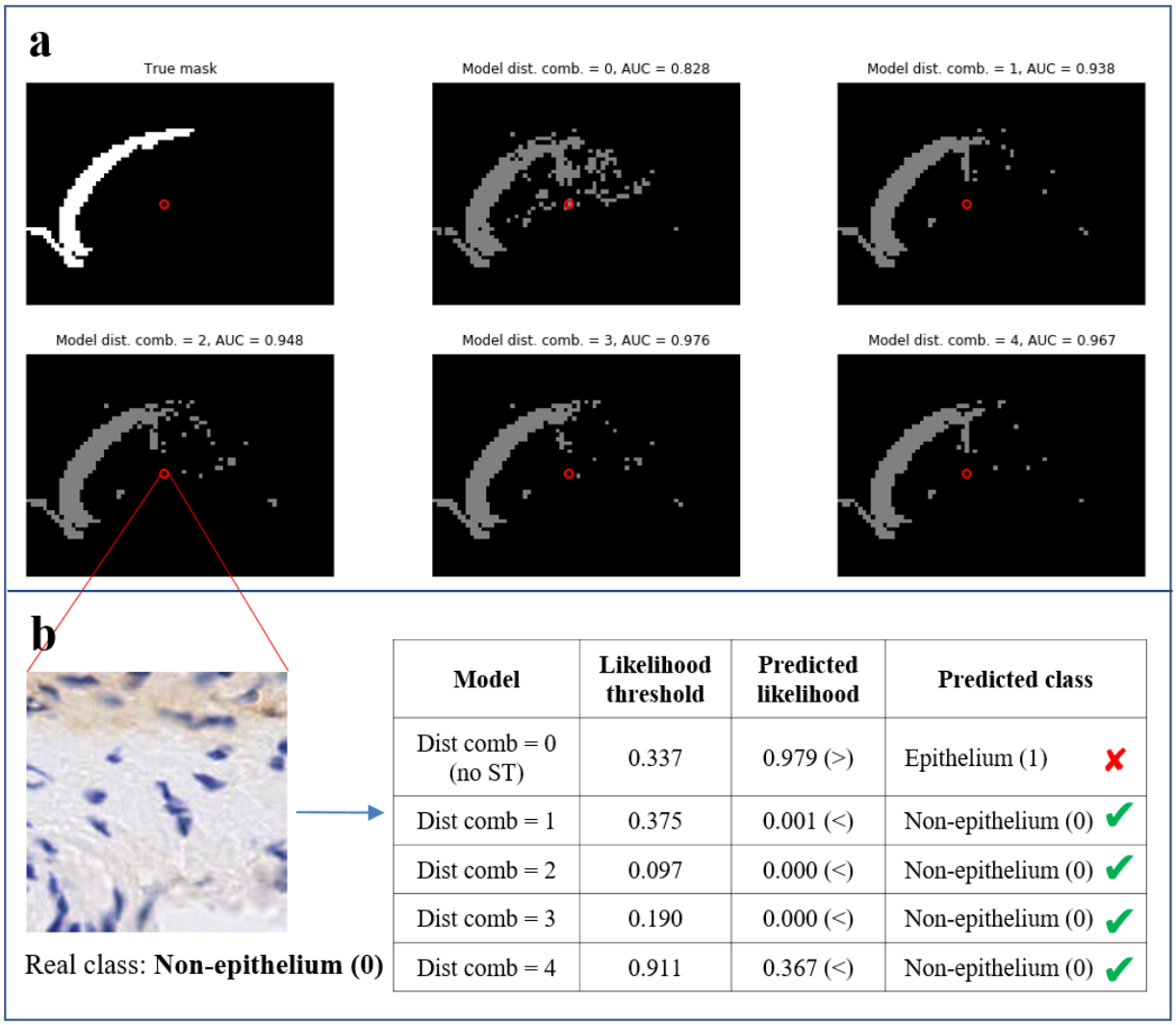


Fig. 4.18. (*top*) ROC curves to compare segmentation performance of models with different values for α on the p16-S360 test set. Models with Stability Training have a significantly higher performance. Performance for α=100, shows a significant decrease. (*bottom*) ROC curves to compare segmentation performance of models with different distortion combinations on the p16-S360 test set. Increasing the level of distortion increases significantly the model segmentation performance for these WSIs.

As observed in Fig. 4.19 (a), models with distortion combination 3 and 4 have the best segmentation performance for this WSI (AUC=0.976 and 0.967 respectively). Once again, the major difference between good and bad performances is related to the number of non-epithelium tiles falsely classified as epithelium, or false negatives. This can be observed clearly in the segmentation from the model with no Stability Training (distortion combination 0) (AUC=0.828), with a large non-epithelium region marked as epithelium. A boxplot of the AUC distribution per WSI can be observed in Appendix B, along with the mean and standard deviation of the AUC. An example tile is also presented in Fig. 4.19 (b), together with the predicted class from the model without Stability Training and the models with different distortion combinations. It can be observed that the model without Stability Training incorrectly predicted the class “epithelium”, while the remaining models correctly classified it as “non-epithelium”.



Model	Likelihood threshold	Predicted likelihood	Predicted class
Dist comb = 0 (no ST)	0.337	0.979 (>)	Epithelium (1) ✘
Dist comb = 1	0.375	0.001 (<)	Non-epithelium (0) ✔
Dist comb = 2	0.097	0.000 (<)	Non-epithelium (0) ✔
Dist comb = 3	0.190	0.000 (<)	Non-epithelium (0) ✔
Dist comb = 4	0.911	0.367 (<)	Non-epithelium (0) ✔

Fig. 4.19. (a) Segmentations of a WSI from the p16-S360 test set, by models with different distortion combinations. The best performing model for this WSI was distortion combination 3 (AUC=0.976), while the worst performing model was distortion combination 0 (no Stability Training) (AUC=0.828). (b) An example tile shows a wrong classification from the model without Stability Training, assigning class “epithelium” to a “non-epithelium” tile, while the models with Stability Training correctly classify it as “non-epithelium”.

4.3.4. Results for CD3/CD8-stained WSIs scanned with NanoZoomer HT (E.3)

The following section describes the results obtained when testing the trained Deep Learning models on WSIs that were scanned with the NanoZoomer-HT but stained with the CD3 and CD8 IHC stains. The difference between both stains is notable, as could be observed in section 3.3, which is why differences were expected between models with and without Stability Straining. The metric used for performance evaluation was the AUC.

E.3.1. CD3-stained WSIs

From a tile-wise perspective, the results obtained for the CD3-stained slides shows that no significant difference exists in the segmentation performance-as can be observed in Fig. 4.20. Even though a slightly lower performance can be observed for the model trained without Stability Training (AUC=0.973), the difference is low. Regarding α, an improvement can be observed as the value of α increases up to α=100, where a decrease in performance is observed. It is interesting to observe, however, that the results for α=100 are not notably different from the other. This contrasts with the results obtained for the WSIs scanned with different scanners, where the performance of the model with α=100 was significantly lower than that of the other α values.

In the case of the distortion combination, there seems to be no correlation between level of distortion and performance. The models with distortion combination 2 and 4 have a lower performance (AUC=0.979 in both cases) than the models with distortion combination 1 and 3 (AUC=0.983 and 0.982 respectively). Nonetheless, all models trained with Stability Training show a better performance than the model with no Stability Training.

The optimal likelihood threshold was calculated using the Youden Index as described in section 3.7. The threshold that maximizes the Youden Index for all tiles in this dataset was selected. With the exception of the model with α=100 (0.672), the calculated threshold of all the other models was between 0.132 and 0.400.

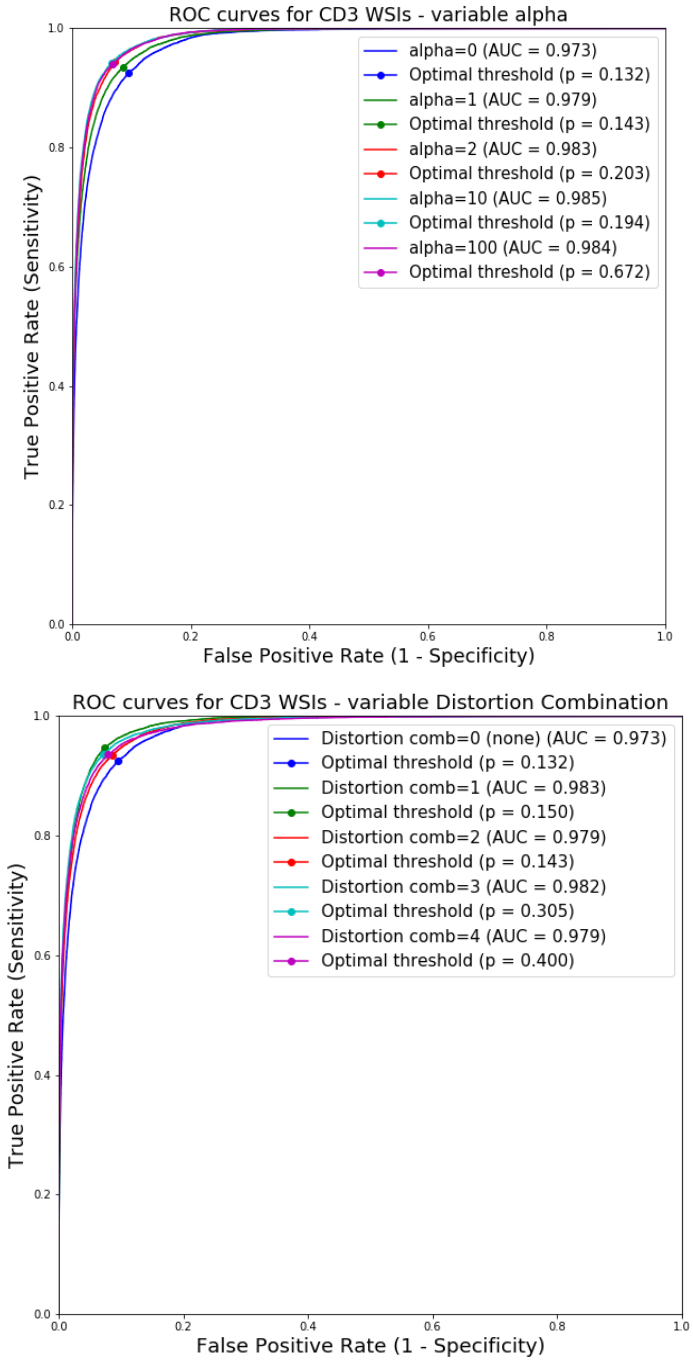
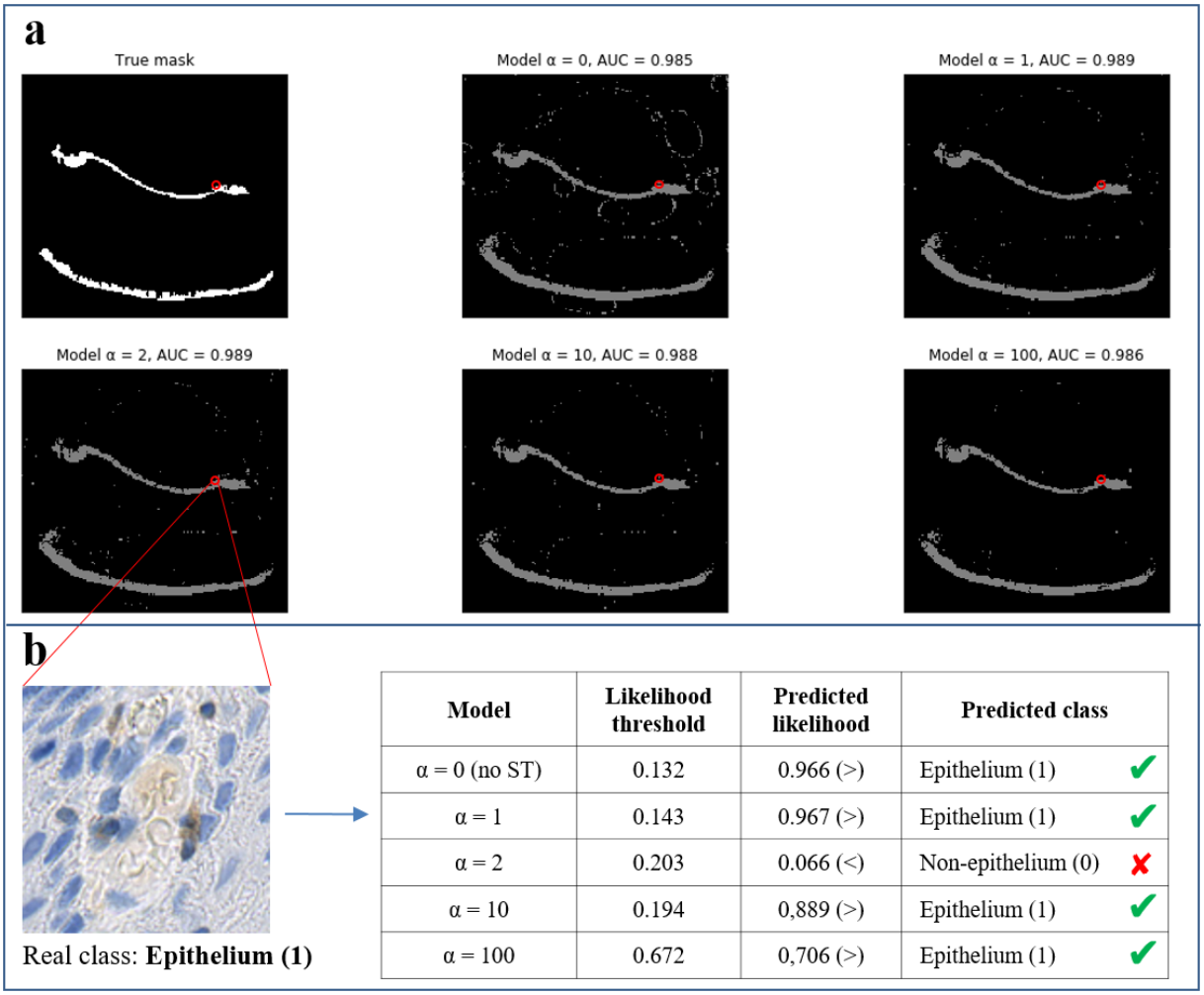


Fig. 4.20. (top) ROC curves to compare segmentation performance of models with different values for α on the CD3-HT test set. A slight improvement can be observed in models with Stability Training (α≠0) with respect to the model with no Stability Training. (bottom) ROC curves to compare segmentation performance of models with different distortion combinations on the CD3-HT test set. The model without Stability Training (distortion comb 0) has the lowest performance.

As can be observed in Fig. 4.21 (a), the model with no Stability Training (α=0) had the lowest performance (AUC=0.985), even though the difference with the other models was not that elevated. However, it can be observed that this segmentation contains some rounds marks which were classified as epithelium. These marks correspond to bubbles in the WSI, and the models trained with Stability Training seem to have been able to correctly recognize them as non-epithelium. The fact that the AUC is similar between all models may be due to the large number of non-epithelium tiles on this WSI. The number of false positives is not high enough to have a big impact on the AUC, even though the bubbles are visually evident in the segmentation from the model without Stability Training (α=0). A boxplot of the AUC distribution per WSI can be observed in Appendix B, along with the mean and standard deviation of the AUC. An example tile is also presented in Fig. 4.21 (b), together with the predicted class from the model without Stability Training and the models with different values of α. It can be observed that all models correctly predicted the class as "epithelium", with the exception of model α. This means that, for this specific tile, the model without Stability Training had a better classification than α=2. This does not necessarily mean that the model with α=2 performs worse, since the overall performance is what determines which model performs better.



Model	Likelihood threshold	Predicted likelihood	Predicted class
α = 0 (no ST)	0.132	0.966 (>)	Epithelium (1) ✔
α = 1	0.143	0.967 (>)	Epithelium (1) ✔
α = 2	0.203	0.066 (<)	Non-epithelium (0) ✘
α = 10	0.194	0,889 (>)	Epithelium (1) ✔
α = 100	0.672	0,706 (>)	Epithelium (1) ✔

Fig. 4.21. (a) Segmentations of a WSI from the CD3-HT test set, by models with different values of α. The worst performing model was α=0 (no Stability Training) (AUC=0.985). Significant noise can be observed in this segmentation, which is mostly caused by artifacts such as bubbles. The models with Stability Training managed to correctly recognize those bubbles as non-epithelium. (b) An example tile shows that all models classified the tile correctly as “epithelium”, with the exception of the model with α=2.

E.3.2. CD8-stained WSIs

From a tile-wise perspective, the results obtained for the CD8-stained slides show that all models trained with Stability Training outperformed the model without Stability Training (AUC=0.961), as can be observed in Fig. 4.22. In this case, the difference is slightly higher than the one observed for the CD3-stained WSIs. Regarding α, the segmentation performance improves as α increases, all the way up to α=10 (AUC=0.980), having a lower performance in α=100 (AUC=0.972). At this point, a pattern has begun to emerge: α=10 is the best performing model in practically all test sets. Regarding the distortion combination, segmentation performance increases as the distortion increases, up to distortion combination 3 (AUC=0.978). However, the difference in performance between these models is not significant.

The optimal likelihood threshold was calculated using the Youden Index as described in section 3.7. The threshold that maximizes the Youden Index for all tiles in this dataset was selected. With the exception of the models with α=100 and distortion combination 4 (0.552 and 0.613 respectively), the calculated threshold of all the other models was between 0.117 and 0.250.

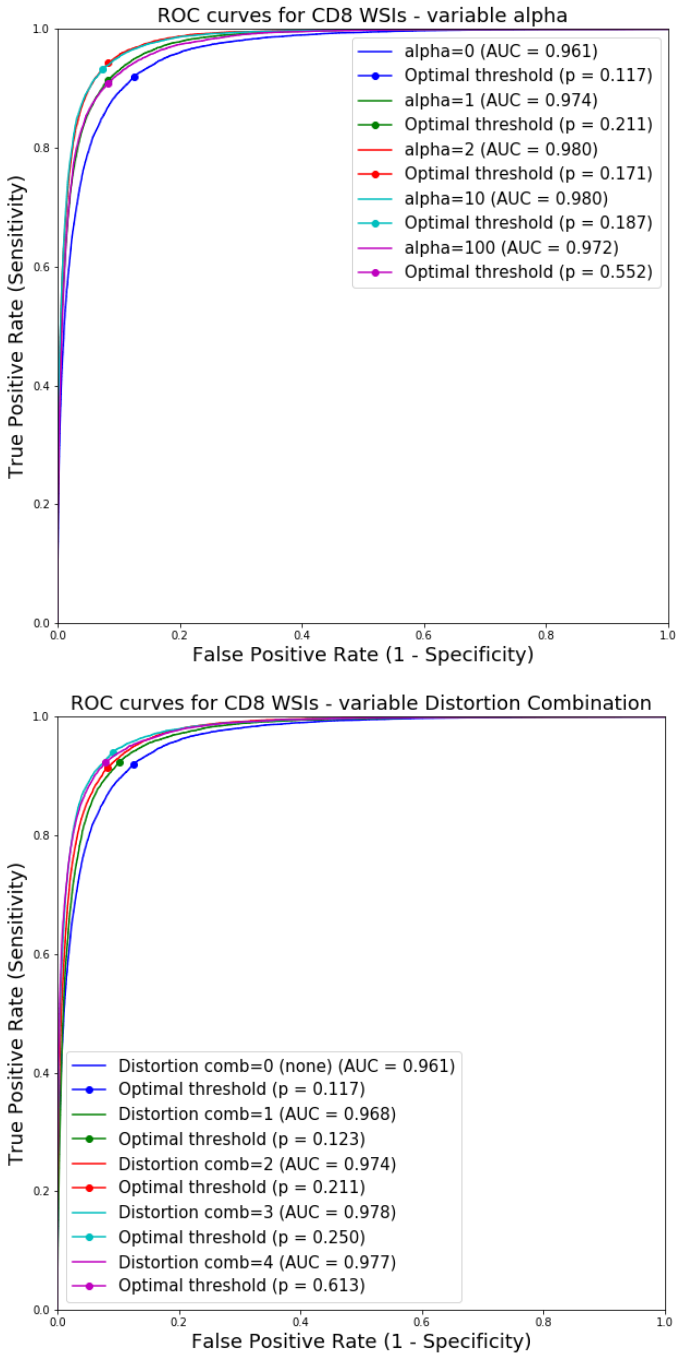
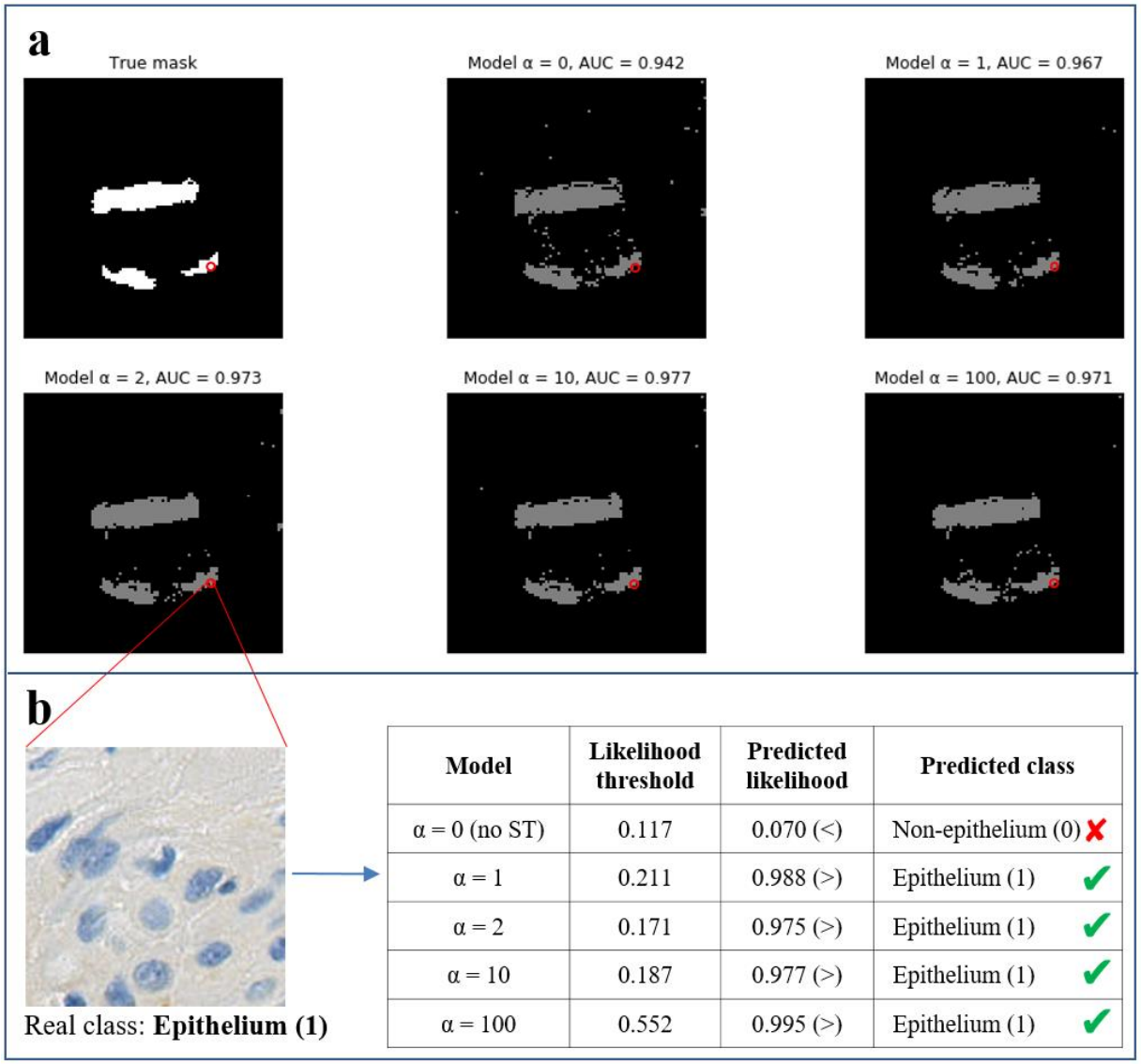


Fig. 4.22. (top) ROC curves to compare segmentation performance of models with different values for α on the CD8-HT test set. A slight improvement can be observed in models with Stability Training (α≠0) with respect to the model with no Stability Training. (bottom) ROC curves to compare segmentation performance of models with different distortion combinations on the CD8-HT test set. The model without Stability Training (distortion comb 0) has the lowest performance.

As can be observed in Fig. 4.23 (a), the model without Stability Training has the lowest performance for this WSI (AUC=0.942). False positives can clearly be seen in the form of grey "island" squares where there should not be (as compared with the true mask). Regarding the models with Stability Training, the segmentations are visually similar. A "dirtier" segmentation can be observed for α=10 (AUC=0.971), which is consistent with the results that have been observed in almost all test sets. A boxplot of the AUC distribution per WSI can be observed in Appendix B, along with the mean and standard deviation of the AUC. An example tile is also presented in Fig. 4.23 (b), together with the predicted class from the model without Stability Training and the models with different values of α. It can be observed that the model without Stability Training incorrectly predicted the class "non-epithelium", while the remaining models correctly classified it as "epithelium".



Model	Likelihood threshold	Predicted likelihood	Predicted class
α = 0 (no ST)	0.117	0.070 (<)	Non-epithelium (0) ✘
α = 1	0.211	0.988 (>)	Epithelium (1) ✔
α = 2	0.171	0.975 (>)	Epithelium (1) ✔
α = 10	0.187	0.977 (>)	Epithelium (1) ✔
α = 100	0.552	0.995 (>)	Epithelium (1) ✔

Fig. 4.23. (a) Segmentations of a WSI from the CD8-HT test set, by models with different values of α. The worst performing model was α=0 (no Stability Training) (AUC=0.942). A slightly higher number of false positives can be observed. (b) An example tile shows a wrong classification from the model without Stability Training, while the model with Stability Training correctly classify it as "epithelium".

5. Discussion

Robustness is fundamental to promote the adoption of clinical decision support systems based on machine learning. In other words, these systems must provide consistent information to the pathologist, regardless of the hardware used to generate the WSIs, the quality of the staining or the technicians handling the equipment. The purpose of this work was to contribute to the field of digital pathology with the assessment of Stability Training as a method to increase the predictive robustness of Deep Learning models. A few publications were found which directly addressed robustness in digital pathology. These applied image processing and Deep Learning-based methods to assess within and between-WSI predictive robustness; however, to the knowledge of the author of this work, Stability Training had not been tested in digital pathology.

In this work, Stability Training was evaluated specifically in the context of epithelium segmentation in cervical histological WSIs. Deep Learning models were trained with different levels of Stability Training (including one model without Stability Training for reference), using WSIs from a specific scanner (NanoZoomer-HT) and with a specific IHC stain (p16). The models were then tested on WSIs from different scanners (NanoZoomer-XR and NanoZoomer S360) and with different IHC stains (CD3 and CD8), in order to evaluate if models with Stability Training had a better performance on these WSIs than the model without Stability Training.

In addition to the evaluation of Stability Training, two previous methodological steps were performed. The first consisted in the removal of background and the processing of the image set (i.e. the set of WSIs) into a dataset for training and testing of the Deep Learning models. The second consisted in selecting a base CNN architecture and a set of optimal hyper-parameter values. These two steps are common to almost any Deep Learning application in digital pathology.

Background removal and dataset preparation

The exploratory analysis of the WSIs was an important step to become familiar with the characteristics of the background, as well as the characteristics of the cervical tissue. It allowed to understand that slides are full of small particles and other elements that need to be considered in the process of removing the background from the dataset. Even though the method (i.e. mode

range + laplacian filter) was simple, it proved to be effective for the image sets used in the project. This does not mean that the method is directly applicable to WSIs from any source, so probably a more robust, generalized background removal method could be explored in order to work with WSIs from different sources.

By selecting the best method for background removal, and through the application of data augmentation and tile labelling, *specific objective 1* was accomplished: *preprocess the image into datasets for Deep Learning model training and evaluation.*

Base architecture selection and hyper-parameter fine-tuning

In general, the InceptionV3 architecture worked as expected, with a reasonable training time and a high segmentation performance. It could be argued that ResNet could have been used because of its slightly better performance, but the use of InceptionV3 architecture proved effective in epithelium segmentation and saved significant time, considering that training the models Stability Training took more than 20 hours each. Nonetheless, the use of multiple GPUs is strongly suggested for future work, as it could significantly speed training time. One thing that was not expected was the performance of the InceptionResNetV2 architecture. Given its depth, it was expected that this network outperformed the others, which did not happen. One possible explanation is the fact that the same hyper-parameters were used for all architectures, while some architectures may function optimally with different hyper-parameter values. It is likely that performance would have improved with a higher learning rate. For future work, further evaluation of hyper-parameters is suggested. Another interesting experiment would be to evaluate model performance for different tile sizes. This variable was originally part of this project, but was finally discarded in order to focus on Stability Training.

By selecting the best performing architecture and hyper-parameter values, *specific objective 2* was accomplished*: Select the optimal base architecture and hyper-parameter values to be used during Stability Training.*

Stability Training and evaluation of robustness

In order to evaluate robustness to image variations using Stability Training, a set of 8 models were trained. These models shared the architecture and hyper-parameters obtained through the fulfillment of *specific objective 2*, and were trained using the datasets obtained through the fulfillment of *specific objective 1*. By training these 8 models and selecting the best performing epoch for each model, *specific objective 3* was accomplished: *train a set of Deep Learning models using different levels of Stability Training on a dataset of WSIs with the same IHC staining and from the same scanner*

The results obtained in the Stability Training section suggest that this method has a positive impact in robustness to images from different scanners and different stains. This does not mean that models with Stability Training performed better than the model without Stability Training on each and every tile; it means that, overall, the models with Stability Training classify more tiles correctly.

Regarding α, the best performing model was α=10 in practically each of the test datasets. However, an excessive weight of the stability component proved to be counterproductive in many cases. This was very clear in the case of α=100, which performed significantly worse and, in one case, even worse than the model without Stability Training. Regarding the distortion combination, all models trained with Stability Training outperformed the model with no Stability Training, in all test sets. In general, the models with a distortion combination of 2 or 3 had a better performance, but distortion combination 4 did not perform significantly worse. This could be explained by the fact that, for high distortion combinations, some of the generated distortions will still be lighter and within the ranges of lower distortion combinations, since these have a random component. Future work should involve the evaluation of different distortion combinations, in order to determine which image properties have a higher positive impact in segmentation performance. The distortion of small regions within tiles instead of the whole tiles could also be explored, since it is frequent to find blurs or color variations only in sections of a tile.

Almost all of the calculated optimal likelihood thresholds are between 0.10 and 0.35. This suggests that models learned to recognize non-epithelium tiles (class 0) more clearly than epithelium tiles (class 1). In other words, non-epithelium tiles tended to have a predicted likelihood closer to 0, while epithelium tiles may have more varied likelihood values (i.e. a likelihood distribution with a higher variance). A possible explanation for this may be the class imbalance between the number of epithelium and non-epithelium tiles. Because the number of epithelium tiles was much smaller, data augmentation was applied on this class. However, it is likely that this method was not sufficient to balance the learning process of the Deep Learning models. Interestingly, some models (especially distortion combination 4 and α=100) tended to have a threshold that was closer to 0.5. This may be explained by a higher variability of the predicted likelihood of non-epithelium tiles.

In E.1, the performance of the models was evaluated on WSIs scanned with the NanoZoomer-HT and stained with the p16 IHC stain (i.e. the same as the WSIs used to train the models). Regarding WSIs scanned in-focus, models with Stability Training performed similarly to the model without Stability Training. This is consistent with previous expectations, since both the training and testing WSIs had the same treatment. Regarding WSIs scanned out-of-focus (blurry), the model without Stability Training performed significantly worse than with WSIs scanned in-focus. The models with Stability Training also performed worse, but the difference was lower, suggesting that these models were more robust to the lack of focus.

In E.2, the performance of the models was evaluated on WSIs scanned with the NanoZoomer-XR and the NanoZoomer S360. In both cases, the model without Stability Training had a significantly lower performance than the models with Stability Training, meaning that the models with Stability Training were more robust to the image variations associated with a different scanner.

In E.3, the performance of the models was evaluated on WSIs scanned with the NanoZoomer-HT (i.e. the same used for training) but stained with the CD3 and CD8 IHC stains instead of the p16 (i.e. the one used for training). In both cases, the model without Stability Training performed

worse than the models with Stability Training, though the difference was not as pronounced as in the case of the WSIs from different scanners.

An interesting observation was the poor performance of the model without Stability Training on the WSIs from different scanners. A better performance was expected, since the difference between these WSIs did not seem significant visually. It was also surprising to see a relatively high performance of the model without Stability Training on the WSIs with different IHC stains, since the differences between the p16 and the CD3/CD8 WSIs was visually very clear. It is possible that all models were able to identify cell morphology regardless of the color (at least to a certain degree), and that the variations associated with the scanner had a much deeper impact in the performance of the model trained without Stability Training. For future work, the evaluation of performance on new scanners is suggested, as this was the context where Stability Training provided the most noticeable improvement in performance. All this being said, the results suggest that *Stability Training does have a positive effect in the segmentation performance for WSIs with different characteristics than the WSIs used for model training*, therefore corroborating the hypothesis. This was true in all of the datasets that were evaluated, including the WSIs from the same "family" that was used to train the models (p16-stained WSIs scanned with the NanoZoomer-HT).

By evaluating the 8 models that were previously trained (*specific objective 3*), *specific objective 4* was accomplished: *Compare the segmentation performance of these models with a model trained without Stability Training on WSIs from different tissue scanners and with different IHC stains.*

There are multiple additional aspects that could be analyzed in future work. One element to include should be the evaluation of different methods to calculate the optimal likelihood threshold. Another element to include is a "post-processing" step to clean the raw segmentations provided by the Deep Learning models. Simple operations such as "dilate" and "erode" could help to remove these "island" false positives, as well as fill the gaps inside the epithelium, which are caused by false negatives.

6. Conclusion

Robustness in Deep Learning is still a challenge to address in multiple fields. The tolerance to heterogeneous data can allow Deep Learning-based applications to scale and be confidently adopted in real world scenarios. This is especially relevant in the medical field, where mistakes can lead to poor diagnoses and, eventually, to the application of incorrect treatments for patients. The main objective of this project was directly linked to this concept, by applying a Deep Learning method focused on robustness to solve the specific problem of segmenting epithelium in histopathological cervical WSIs.

This work proposes a possible pipeline for the application of Deep Learning in digital pathology, starting with the processing of a set of WSIs into a usable dataset, and ending with a performance analysis focused on robustness to heterogeneity. A simple but effective method for background analysis and removal was proposed, which could be applicable in any application using WSIs. A comparison between different Deep Learning architectures and hyper-parameter values was performed to build the optimal Deep Learning model, following common practices in parameter fine-tuning. More importantly, Stability Training was applied for the first time in digital pathology to train models that are more robust to heterogeneous WSIs, with results validating it as an effective method.

Digital pathology is an emerging field with enormous potential to change the way conventional pathology is practiced. However, and as with any new technology, the community is in the process of adapting to its use, just as digital pathology is adapting to the requirements of the community. In the information-driven era we are living in, the large number of WSIs that are being stored constitutes an enormous opportunity to apply innovative techniques in the field of Deep Learning, which can have a positive impact in the practice of medicine. Nonetheless, researchers in the field have the duty to guarantee that this technology is reliable and safe, and on this regard, robustness is key.

7. References

[1] S. P. Bhavnani, J. Narula, and P. P. Sengupta, “Mobile technology and the digitization of healthcare,” *Eur. Heart J.*, vol. 37, no. 18, pp. 1428–1438, May 2016.

[2] W. Raghupathi and V. Raghupathi, “Big data analytics in healthcare: promise and potential,” *Heal. Inf. Sci. Syst.*, vol. 2, no. 1, pp. 1–10, Dec. 2014.

[3] J. Archenaa and E. A. M. Anita, “A survey of big data analytics in healthcare and government,” in *Procedia Computer Science*, 2015, vol. 50, pp. 408–413.

[4] G. Litjens *et al.*, “A survey on deep learning in medical image analysis.,” *Med. Image Anal.*, vol. 42, pp. 60–88, 2017.

[5] F. Bray, J. Ferlay, I. Soerjomataram, R. L. Siegel, L. A. Torre, and A. Jemal, “Global cancer statistics 2018: GLOBOCAN estimates of incidence and mortality worldwide for 36 cancers in 185 countries,” *CA. Cancer J. Clin.*, vol. 68, no. 6, pp. 394–424, Nov. 2018.

[6] X. Castellsagué, “Natural history and epidemiology of HPV infection and cervical cancer,” *Gynecol. Oncol.*, vol. 110, no. 3 SUPPL.2, pp. S4–S7, Sep. 2008.

[7] M. Schiffman, P. E. Castle, J. Jeronimo, A. C. Rodriguez, and S. Wacholder, “Human papillomavirus and cervical cancer,” *Lancet*, vol. 370, no. 9590, pp. 890–907, Sep. 2007.

[8] D. M. Parkin and F. Bray, “The burden of HPV-related cancers,” *Vaccine*, vol. 24, no. SUPPL. 3, pp. S11–S25, Aug. 2006.

[9] L. S. Massad *et al.*, “2012 updated consensus guidelines for the management of abnormal cervical cancer screening tests and cancer precursors,” *Obstetrics and Gynecology*, vol. 121, no. 4. pp. 829–846, Apr-2013.

[10] D. R. Lowy and J. T. Schiller, “Prophylactic human papillomavirus vaccines,” *J. Clin. Invest.*, vol. 116, no. 5, pp. 1167–1173, May 2006.

[11] A. G. Ostör, “Natural history of cervical intraepithelial neoplasia: a critical review.,” *Int. J. Gynecol. Pathol.*, vol. 12, no. 2, pp. 186–92, Apr. 1993.

[12] S. Wang and W. Wu, “DNA Methylation Alterations in Human Cancers,” in *Epigenetics in Human Disease*, Academic Press, 2018, pp. 109–139.

[13] N. Wentzensen, M. Schiffman, T. Palmer, and M. Arbyn, “Triage of HPV positive women in cervical cancer screening,” *J. Clin. Virol.*, vol. 76, pp. S49–S55, Mar. 2016.

[14] E. H. Hopman, F. J. Voorhorst, P. Kenemans, C. J. L. M. Meyer, and T. J. M.

Helmerhorst, “Observer Agreement on Interpreting Colposcopic Images of CIN,” *Gynecol. Oncol.*, vol. 58, no. 2, pp. 206–209, Aug. 1995.

[15] M. H. Stoler *et al.*, “The interpretive variability of cervical biopsies and its relationship to HPV status,” *Am. J. Surg. Pathol.*, vol. 39, no. 6, pp. 729–736, 2015.

[16] K. Kaliyappan, M. Palanisamy, J. Duraiyan, and R. Govindarajan, “Applications of immunohistochemistry,” *J. Pharm. Bioallied Sci.*, vol. 4, no. 6, p. 307, 2012.

[17] R. Klaes *et al.*, “p16INK4a immunohistochemistry improves interobserver agreement in the diagnosis of cervical intraepithelial neoplasia,” *Am. J. Surg. Pathol.*, vol. 26, no. 11, pp. 1389–1399, Nov. 2002.

[18] M. Serrano, G. J. Hannon, and D. Beach, “A new regulatory motif in cell-cycle control causing specific inhibition of cyclin D/CDK4,” *Nature*, vol. 366, no. 6456, pp. 704–707, 1993.

[19] K. Nehls *et al.*, “p16 methylation does not affect protein expression in cervical carcinogenesis,” *Eur. J. Cancer*, vol. 44, no. 16, pp. 2496–2505, Nov. 2008.

[20] R. Chen, Y. Gong, D. Zou, L. Wang, L. Yuan, and Q. Zhou, “Correlation between subsets of tumor-infiltrating immune cells and risk stratification in patients with cervical cancer,” *PeerJ*, vol. 2019, no. 10, 2019.

[21] J. Galon *et al.*, “Type, density, and location of immune cells within human colorectal tumors predict clinical outcome,” *Science (80-.).*, vol. 313, no. 5795, pp. 1960–1964, Sep. 2006.

[22] S. Michel *et al.*, “High density of FOXP3-positive T cells infiltrating colorectal cancers with microsatellite instability,” *Br. J. Cancer*, vol. 99, no. 11, pp. 1867–1873, Dec. 2008.

[23] N. Halama *et al.*, “Localization and density of immune cells in the invasive margin of human colorectal cancer liver metastases are prognostic for response to chemotherapy,” *Cancer Res.*, vol. 71, no. 17, pp. 5670–5677, Sep. 2011.

[24] N. Halama *et al.*, “Quantification of prognostic immune cell markers in colorectal cancer using whole slide imaging tumor maps,” *Anal Quant Cytol Histol*, vol. 32, no. 6, pp. 333–40, 2010.

[25] B. Chen, L. Liu, H. Xu, Y. Yang, L. Zhang, and F. Zhang, “Effectiveness of immune therapy combined with chemotherapy on the immune function and recurrence rate of cervical cancer,” *Exp. Ther. Med.*, vol. 9, no. 3, pp. 1063–1067, 2015.

[26] A. M. Heeren *et al.*, “Neoadjuvant cisplatin and paclitaxel modulate tumor-infiltrating T cells in patients with cervical cancer,” *Cancer Immunol. Immunother.*, vol. 68, no. 11, pp. 1759–1767, Nov. 2019.

[27] Y. L. Woo *et al.*, “Characterising the local immune responses in cervical intraepithelial neoplasia: A cross-sectional and longitudinal analysis,” *BJOG An Int. J. Obstet. Gynaecol.*, vol. 115, no. 13, pp. 1616–1622, Dec. 2008.

[28] F. Zennaro *et al.*, “Digital Radiology to Improve the Quality of Care in Countries with Limited Resources: A Feasibility Study from Angola,” *PLoS One*, vol. 8, no. 9, p. e73939, Sep. 2013.

[29] L. Pantanowitz, N. Farahani, and A. Parwani, “Whole slide imaging in pathology: advantages, limitations, and emerging perspectives,” *Pathol. Lab. Med. Int.*, vol. 7, p. 23, Jun. 2015.

[30] “Cytomine: Free Open source Web-based Digital Pathology (WSI) solution with Machine learning flavor.” [Online]. Available: https://medevel.com/cytomine-digital-pathology/. [Accessed: 17-Jul-2020].

[31] A. B. Farris, C. Cohen, T. E. Rogers, and G. H. Smith, “Whole Slide Imaging for Analytical Anatomic Pathology and Telepathology: Practical Applications Today, Promises, and Perils,” *Arch. Pathol. Lab. Med.*, vol. 141, no. 4, pp. 542–550, Apr. 2017.

[32] J. Molin, S. Thorstenson, and C. Lundström, “Implementation of large-scale routine diagnostics using whole slide imaging in Sweden: Digital pathology experiences 2006-2013,” *J. Pathol. Inform.*, vol. 5, no. 1, p. 14, 2014.

[33] P. S. Nielsen, J. Lindebjerg, J. Rasmussen, H. Starklint, M. Waldstrøm, and B. Nielsen, “Virtual microscopy: An evaluation of its validity and diagnostic performance in routine histologic diagnosis of skin tumors,” *Hum. Pathol.*, vol. 41, no. 12, pp. 1770–1776, Dec. 2010.

[34] A. Al Habeeb, D. Ghazarian, and A. Evans, “Virtual microscopy using whole-slide imaging as an enabler for teledermatopathology: A paired consultant validation study,” *J. Pathol. Inform.*, vol. 3, no. 1, p. 2, 2012.

[35] “Digital Pathology Market Size,” 2020. [Online]. Available: https://www.grandviewresearch.com/industry-analysis/digital-pathology-systems-market. [Accessed: 13-Feb-2020].

[36] W. S. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity,” *Bull. Math. Biophys.*, vol. 5, no. 4, pp. 115–133, Dec. 1943.

[37] B. Widrow and M. Hoff, “Adaptive switching circuits (No. TR-1553-1),” in *Stanford Univ Ca Stanford Electronics Labs*, 1960.

[38] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998.

[39] K. Fukushima, “Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position-Neocognitron-,” *IEICE Tech. Report, A*, vol. 62, no. 10, pp. 658–665, 1979.

[40] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” *Adv. Neural Inf. Process. Syst.*, pp. 1097–1105, 2012.

[41] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016, vol. 2016–Decem, pp. 779–788.

[42] D. Silver *et al.*, “Mastering the game of Go with deep neural networks and tree search,” *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.

[43] A. I. Maqueda, A. Loquercio, G. Gallego, N. Garcia, and D. Scaramuzza, “Event-Based Vision Meets Deep Learning on Steering Prediction for Self-Driving Cars,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5419–5427.

[44] T. Karras, S. Laine, and T. Aila, “A Style-Based Generator Architecture for Generative Adversarial Networks,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2019–June, pp. 4396–4405, Dec. 2018.

[45] A. Esteva *et al.*, “Dermatologist-level classification of skin cancer with deep neural networks,” *Nature*, vol. 542, no. 7639, pp. 115–118, Feb. 2017.

[46] Z. Cheng, Q. Yang, and B. Sheng, “Deep colorization,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, vol. 2015 Inter, pp. 415–423.

[47] J. Deng, W. Dong, R. Socher, L.-J. Li, Kai Li, and Li Fei-Fei, “ImageNet: A large-scale hierarchical image database,” 2010, pp. 248–255.

[48] H. Wang and B. Raj, “On the Origin of Deep Learning,” Feb. 2017.

[49] S. Zheng, Y. Song, T. Leung, and I. Goodfellow, “Improving the robustness of deep

neural networks via stability training," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016, vol. 2016–Decem, pp. 4480–4488.

[50] J. Xu, X. Luo, G. Wang, H. Gilmore, and A. Madabhushi, "A Deep Convolutional Neural Network for segmenting and classifying epithelial and stromal regions in histopathological images," *Neurocomputing*, vol. 191, pp. 214–223, May 2016.

[51] Y. Song, L. Zhang, S. Chen, D. Ni, B. Lei, and T. Wang, "Accurate segmentation of cervical cytoplasm and nuclei based on multiscale convolutional network and graph partitioning," *IEEE Trans. Biomed. Eng.*, vol. 62, no. 10, pp. 2421–2433, Oct. 2015.

[52] D. R. Martin, J. A. Hanson, R. R. Gullapalli, F. A. Schultz, A. Sethi, and D. P. Clark, "A Deep Learning Convolutional Neural Network Can Recognize Common Patterns of Injury in Gastric Pathology," *Arch. Pathol. Lab. Med.*, p. arpa.2019-0004-OA, Jun. 2019.

[53] D. Wang, A. Khosla, R. Gargeya, H. Irshad, and A. H. Beck, "Deep Learning for Identifying Metastatic Breast Cancer," Jun. 2016.

[54] E. Kim, M. Corte-Real, and Z. Baloch, "A deep semantic mobile application for thyroid cytopathology," 2016, vol. 9789, p. 97890A.

[55] X. Zhang, F. Xing, H. Su, L. Yang, and S. Zhang, "High-throughput histopathological image analysis via robust cell segmentation and hashing," *Med. Image Anal.*, vol. 26, no. 1, pp. 306–315, Dec. 2015.

[56] F. Xing and L. Yang, "Robust nucleus/cell detection and segmentation in digital pathology and microscopy images: A comprehensive review," *IEEE Reviews in Biomedical Engineering*, vol. 9. Institute of Electrical and Electronics Engineers, pp. 234–263, 2016.

[57] R. Therrien and S. Doyle, "Role of training data variability on classifier performance and generalizability," in *SPIE*, 2018, vol. 10581, p. 5.

[58] M. S. Hosseini *et al.*, "Atlas of Digital Pathology: A Generalized Hierarchical Histological Tissue Type-Annotated Database for Deep Learning," 2019.

[59] B. S. Veeling, J. Linmans, J. Winkens, T. Cohen, and M. Welling, "Rotation equivariant CNNs for digital pathology," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2018, vol. 11071 LNCS, pp. 210–218.

[60] M. Balkenhol, N. Karssemeijer, G. J. S. Litjens, J. van der Laak, F. Ciompi, and D. Tellez, "H&E stain augmentation improves generalization of convolutional networks for histopathological mitosis detection," in *Medical Imaging 2018: Digital Pathology*, 2018, vol. 10581, p. 34.

[61] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural Networks*, vol. 12, no. 1, pp. 145–151, Jan. 1999.

[62] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proceedings of the 26th International Conference On Machine Learning, ICML 2009*, 2009, pp. 609–616.

[63] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

[64] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2015, vol. 07–12–June, pp. 1–9.

[65] K. Yamaguchi, K. Sakamoto, T. Akabane, and Y. Fujimoto, "A neural network for speaker-independent isolated word recognition," in *First International Conference on Spoken Language Processing*, 1990.

[66] S. Mittal, "A survey of FPGA-based accelerators for convolutional neural networks," *Neural Comput. Appl.*, vol. 32, no. 4, pp. 1109–1139.

[67] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016, vol. 2016–Decem, pp. 770–778.

[68] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016, vol. 2016–Decem, pp. 2818–2826.

[69] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning," *Thirty-First AAAI Conf. Artif. Intell.*, Feb. 2016.

[70] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in

Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2016, vol. 9908 LNCS, pp. 630–645.

[71] J. Laermann, W. Samek, and N. Strodthoff, "Achieving Generalizable Robustness of Deep Neural Networks by Stability Training," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2019, vol. 11824 LNCS, pp. 360–373.

[72] E. A. Freeman and G. G. Moisen, "A comparison of the performance of threshold criteria for binary classification in terms of predicted prevalence and kappa," *Ecol. Modell.*, vol. 217, no. 1–2, pp. 48–58, Sep. 2008.

[73] M. Bekkar, H. K. Djemaa, and T. A. Alitouche, "Evaluation Measures for Models Assessment over Imbalanced Data Sets," *J. Inf. Eng. Appl.*, vol. 3, no. 10, pp. 27–38, 2013.

[74] W. J. Youden, "Index for rating diagnostic tests," *Cancer*, vol. 3, no. 1, pp. 32–35, Jan. 1950.

[75] N. Wentzensen *et al.*, "Accuracy and Efficiency of Deep-Learning-Based Automation of Dual Stain Cytology in Cervical Cancer Screening," *JNCI J. Natl. Cancer Inst.*, 2020.

Appendix

Appendix A: Methodology to process a set of WSIs into a dataset for model training

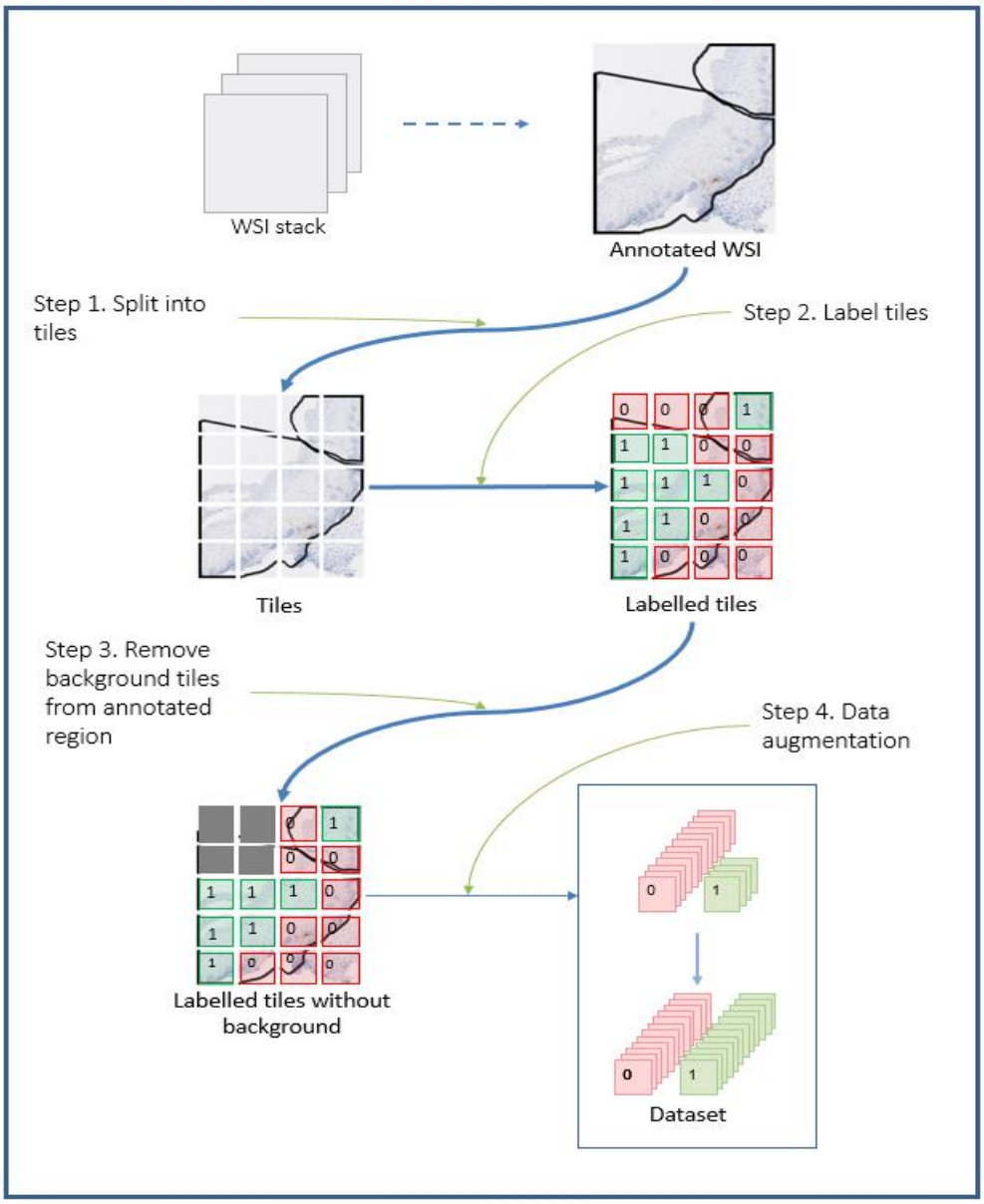


Fig. A.1. Methodology to construct the dataset that will later be used for training. Step 1. The original WSI is split into tiles. Step 2. Tiles are labelled as 1 ("epithelium") or 0 ("non-epithelium") depending on whether these are inside or outside the annotated region, respectively. Partially labelled tiles are labelled as 0. Step 3. Tiles corresponding to background are removed from the set. Step 4. Data augmentation is applied to the smallest class to solve class imbalance.

Appendix B: Analysis of AUC per WSI

Dataset	α	α μ AUC	α σ AUC	Distortion Combination distortion	μ AUC	σ AUC
CD3	0	0.949	0.041	0	0.949	0.041
	1	0.962	0.034	1	0.966	0.032
	2	0.968	0.029	2	0.962	0.034
	10	0.972	0.027	3	0.966	0.034
	100	0.973	0.027	4	0.961	0.041
CD8	0	0.924	0.051	0	0.924	0.051
	1	0.951	0.040	1	0.940	0.045
	2	0.961	0.030	2	0.951	0.040
	10	0.964	0.026	3	0.958	0.031
	100	0.954	0.032	4	0.957	0.033
p16_HT (or p16)	0	0.963	0.031	0	0.963	0.031
	1	0.971	0.026	1	0.974	0.025
	2	0.973	0.025	2	0.971	0.026
	10	0.975	0.025	3	0.970	0.029
	100	0.975	0.025	4	0.965	0.031
p16_s360	0	0.760	0.114	0	0.760	0.114
	1	0.872	0.066	1	0.884	0.072
	2	0.906	0.054	2	0.872	0.066
	10	0.928	0.043	3	0.935	0.043
	100	0.875	0.076	4	0.923	0.055
p16_XR	0	0.888	0.054	0	0.888	0.054
	1	0.924	0.045	1	0.899	0.064
	2	0.941	0.037	2	0.924	0.045
	10	0.941	0.038	3	0.945	0.034
	100	0.864	0.084	4	0.938	0.039
p16blurred	0	0.841	0.096	0	0.841	0.096
	1	0.887	0.088	1	0.876	0.076
	2	0.881	0.106	2	0.887	0.088
	10	0.904	0.105	3	0.890	0.079
	100	0.867	0.102	4	0.908	0.066

Table B.1. Mean (μ) and standard deviation (σ) of the AUC per WSI. The best mean AUC is highlighted green and the worst is highlighted in red. p16_HT is the test set with the same WSI type as the training set. Model α=0 is the same as model distortion=0, and corresponds to the model without Stability Training.

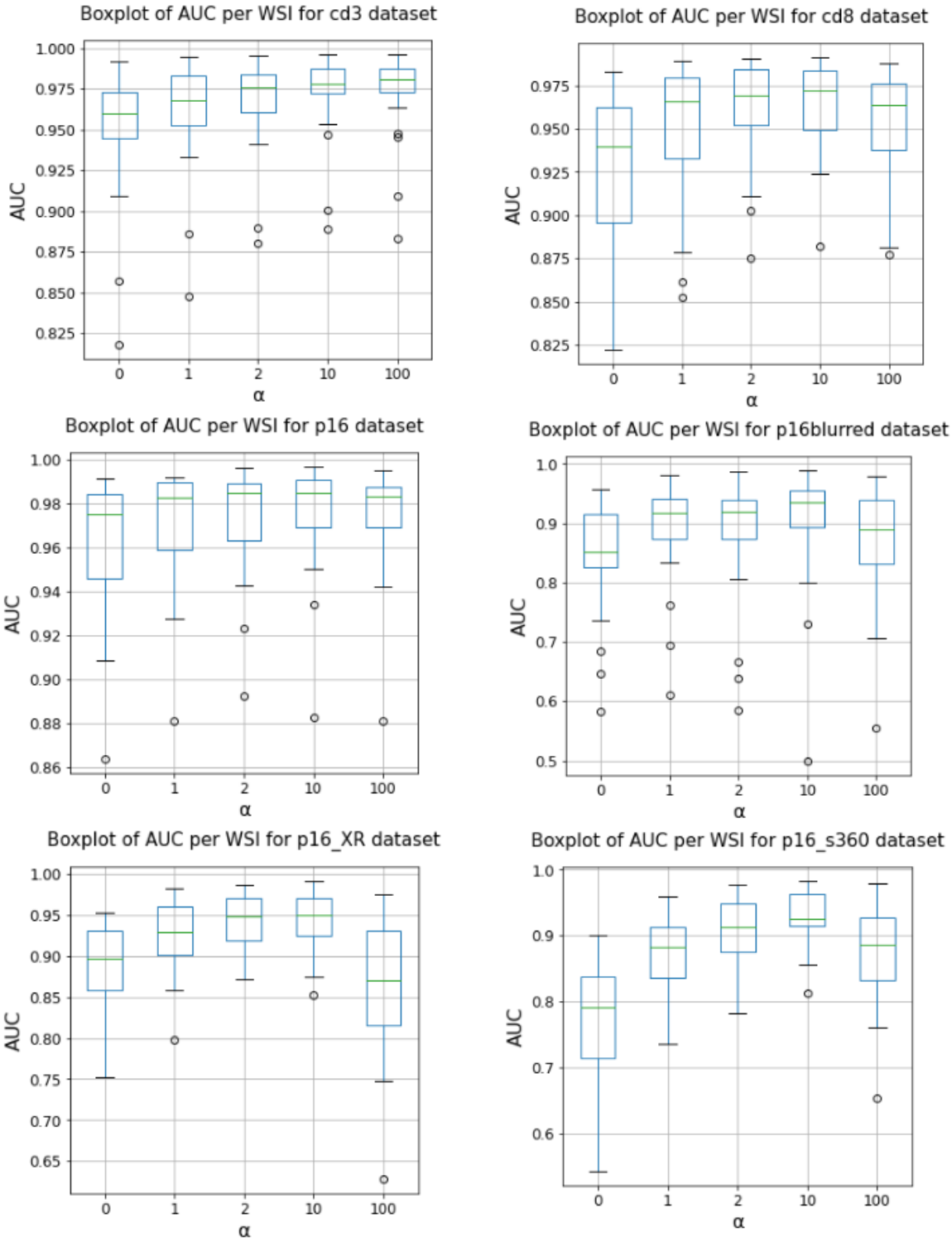


Fig. B.1 Boxplot of AUC per WSI for each test dataset – parameter α. α=0 is the model without Stability Training. The green line dividing each triangle represents the median.

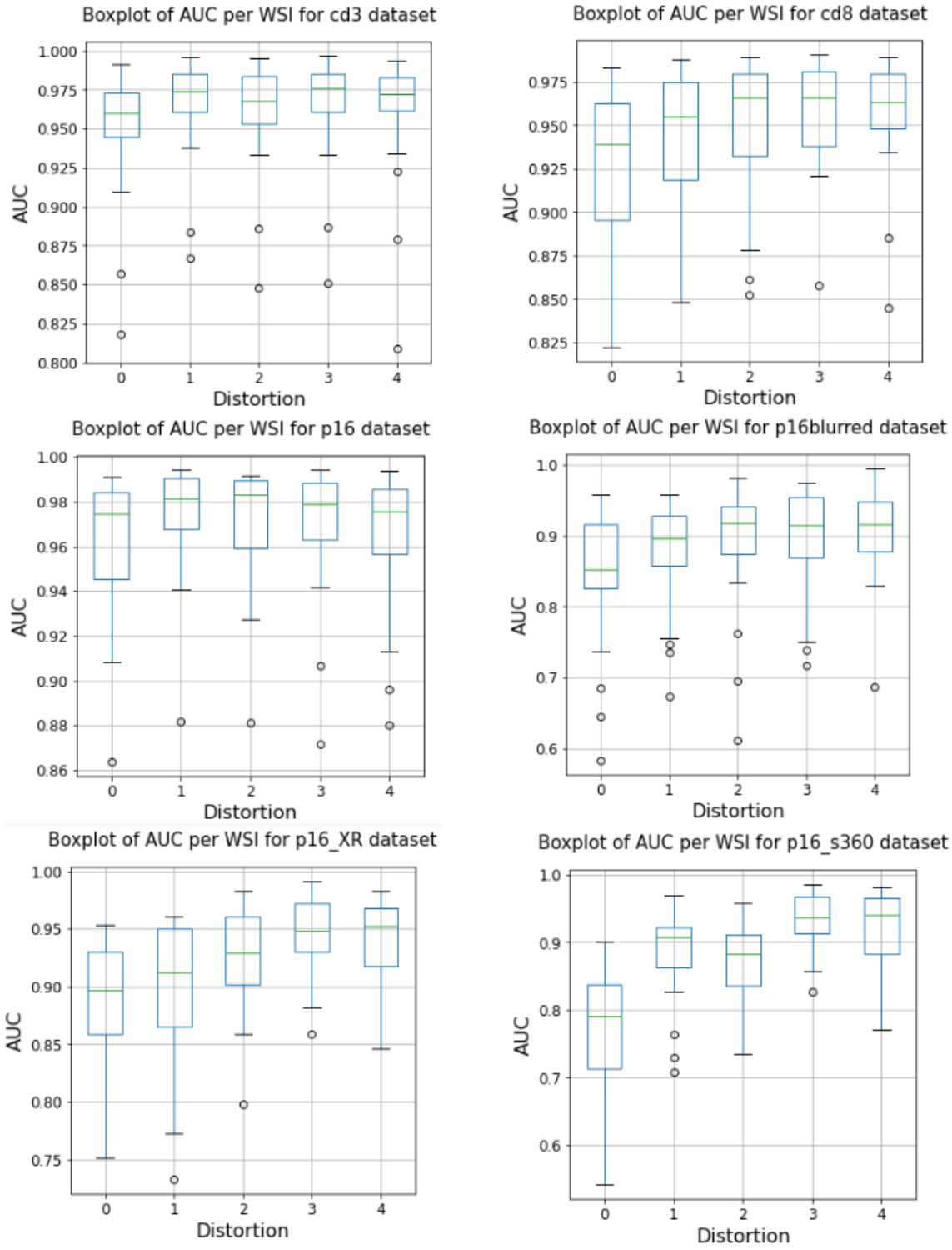


Fig. B.2 Boxplot of AUC per WSI for each test dataset – parameter distortion combination. Distortion 0 is the model without Stability Training. The green line dividing each rectangle represents the median.