Friedrich-Alexander-Universität
Technische Fakultät

Refinement of Whole Slide Image Semantic Segmentation using Weak Regional Supervision

Optimierung der semantischen Segmentierung von Objektträger-Aufnahmen mithilfe von schwacher regionaler Überwachung

Master's Thesis in Computer Science

at the Juniorprofessur für Artificial Intelligence in Medical Imaging
Department Artificial Intelligence in Biomedical Engineering
Friedrich-Alexander-Universität Erlangen-Nürnberg

Supervisor:	Prof. Dr.-Ing. Katharina Breininger, M.Sc. Jingna Qiu, Dr.-Ing. André Aichert
Presented by:	Kirill Menke Kantstr. 13c 91074 Herzogenaurach 22238658
Submission:	27th July 2023

Declaration of Academic Integrity

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

I affirm that I have completed the work without outside help and without using sources other than those given and that the work in the same or a similar form has not been submitted to any other examining authority and has been accepted by them as part of an examination. All explanations that have been taken over literally or analogously are marked as such.

Erlangen, 25th June 2023

Kirill Menke

Abstract

Deep learning-based algorithms show great potential for segmenting different tissue types within whole slide images (WSIs) and are often a prerequisite for other diagnostic computer vision tasks in digital pathology. To reduce the annotation effort and cost of training semantic segmentation models, weakly supervised semantic segmentation (WSSS) algorithms are often applied, which can cope with imprecise or incomplete training labels. Most WSSS methods for histopathological images produce relatively low-resolution segmentation or suffer from oversegmentation. However, an accurate high-resolution segmentation can be the basis for a reliable and robust downstream analysis for disease prognosis or classification. Therefore, this thesis investigates the WSSS approach SwinMIL for high-resolution WSIs and the effect of the pretraining strategy DINO on its segmentation performance. For this purpose, several experiments were conducted and compared in terms of quantitative and qualitative segmentation performance. SwinMIL showed promising potential on high-resolution WSIs, achieving a high level of detail. The initially high false positive rate for epithelial tissue in healthy slides was significantly reduced by applying hard negative mining. However, SwinMIL was still prone to oversegmentation in the presence of cancerous tissue, indicating the need for further refinement. In addition, the weighting of different stages in SwinMIL was found to play a key role in balancing high- and low-level information, which significantly affects pixel-level performance. Contrary to expectations, pretraining with DINO on histopathological image data did not improve the segmentation performance of SwinMIL.

Zusammenfassung

Deep Learning basierte Algorithmen haben ein großes Potential bei der Segmentierung verschiedener Gewebetypen in Objektträger-Aufnahmen und sind oft eine wichtige Voraussetzung für weitere diagnostische Computer Vision Anwendungen in der digitalen Pathologie. Um den Annotationsaufwand und die Kosten für das Training solcher semantischer Segmentierungsmodelle zu reduzieren, werden häufig schwach überwachte semantische Segmentierungsalgorithmen (sogenannte WSSS-Algorithmen) verwendet, die mit ungenauen oder unvollständigen Trainingsannotationen umgehen können. Die meisten WSSS-Verfahren für histopathologische Bilder erzeugen relativ niedrig aufgelöste Segmentierungsmasken oder neigen zur Übersegmentierung. Eine präzise hochauflösende Segmentierung kann jedoch die Grundlage für eine zuverlässige und robuste Analyse zur Vorhersage oder Klassifizierung von Krankheiten bilden. Daher wird in dieser Arbeit das WSSS-Verfahren SwinMIL für hochaufgelöste Objektträger-Aufnahmen und der Einfluss der Pretraining-Strategie DINO auf dessen Segmentierungsperformance untersucht. Dazu wurden mehrere Experimente durchgeführt und hinsichtlich ihrer quantitativen und qualitativen Segmentierungsperformance verglichen. SwinMIL zeigte ein hohes Potential bei hochauflösenden Objektträger-Aufnahmen und erreichte einen hohen Detailgrad. Allerdings neigte SwinMIL immer noch zur Übersegmentierung in der Nähe von Krebsgewebe, was darauf hindeutet, dass weitere Optimierungen erforderlich sind. Darüber hinaus wurde festgestellt, dass die Gewichtung der verschiedenen Stages in SwinMIL eine zentrale Rolle bei der Balance zwischen High- und Low-Level-Informationen spielt und die Performance auf Pixelebene erheblich beeinflusst. Entgegen den urpsrünglichen Erwartungen hat das Pretraining mit DINO mit histopathologischen Bilddaten die Segmentierungsperformance von SwinMIL nicht verbessert.

Contents

1 Introduction

Cancer is a term for a group of diseases that are characterized by the uncontrolled growth and spread of abnormal cells in the body [40]. Normally, cells grow, divide, and die in an orderly fashion. Cancer, however, disrupts this process. This can lead to the formation of abnormal tissue masses called tumors, and cancer cells can spread to other body parts via lymph or blood vessels, a process referred to as metastasis. When cancer is detected at an early stage, it is often easier to treat and the chances of cure are highest [64]. There are several approaches to diagnosing cancer, but one of the most reliable methods is the collection of tissue samples from the body part in question through biopsy followed by chemical staining and histological analysis [63]. Tissue staining is essential before the analysis because otherwise, different components would be difficult to distinguish under the microscope. Among the most important in cancer diagnostics are hematoxylin-eosin (HE) staining and immunohistochemical (IHC) staining [61]. To create a diagnosis, pathologists examine the spatial arrangement and structure of cells, and their assessment regarding tumor grade and stage often decides about the subsequent treatment choices [18]. In this regard, digital pathology opened up new possibilities and has become a rapidly advancing field that is changing the way medical professionals diagnose diseases [22]: Tissue specimens on glass slides are now digitized with high-resolution image scanners and pathologists are moving beyond viewing tissue samples under a microscope to computer-aided analysis via digital whole-slide images (WSIs). This allows for a more efficient workflow. Once scanned, WSIs can be analyzed and managed on the computer screen. Pathologists can navigate faster between specific areas at different zoom levels on the sample and label tissue sections more easily. In addition, digital WSIs open up new possibilities for collaborations and remote diagnostics [22]: Pathologists can easily share histopathology images with their colleagues or other specialists around the world when a rare or complex diagnosis is needed.

Another major advantage offered by digital pathology is the capability of automated histological analysis through computer software. Especially deep learning-based computer vision algorithms have achieved good results in the analysis of WSIs and therefore seem to be the most promising [19]. They are able to identify subtle patterns and features that may be missed by the human eye enabling pathologists to perform a more accurate and reliable diagnosis [45]. Furthermore, the visualization of regions of interest can reduce the slide screening time of pathologists. This is particularly valuable for large giga-pixel WSIs since image sizes in the range of $100,000 \times 100,000$ pixels are quite common [5, 52]. Different tissue types can be recognized and isolated through segmentation to identify specific regions of interest. This thesis specifically focuses on applying deep learning for the semantic segmentation of tumorous tissue on WSIs.

There are numerous deep learning algorithms for the semantic segmentation of images [5, 82]. However, they have been predominantly developed using images of natural scenes and it is

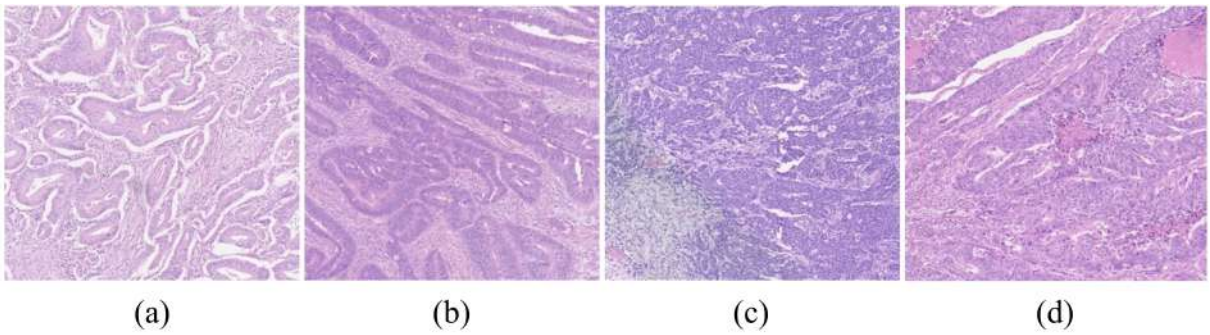


Figure 1.1: Sections of HE stained tissue with different grades of colorectal cancer tumors. (a) and (b) show images of low-grade tumors while (c) and (d) show high-grade tumors.[1]

generally not clear how well their performance translates to image domains with other characteristics [13]. Compared to natural scene images, histopathology images can be more complex. They can contain more intricate structures and finer-grained objects, and the morphological appearance between foreground and background (e.g. between tumor and no tumor) may be less significant or more subtle [13]. Moreover, in the case of cancer, tumor regions are often disconnected and exhibit various morphologies (see Figure 1.1). These circumstances can complicate semantic segmentation and lead to poor performance when algorithms developed on natural scene images are applied to histopathological images. Simply retraining such models on histopathological images is often not sufficient to achieve good performance, and algorithms adapted to the histopathological image domain are usually superior [13].

The most successful models in the histopathology domain have in common that they process a WSI sequentially [19]. In most cases, WSIs are divided into patches and individually fed into the neural network. But there are also other solutions such as partitioning the WSI into superpixels [3]. Approaches in which WSIs are downscaled and processed as a whole generally perform worse [19]. The highest quality in semantic segmentation is achieved by supervised approaches where for each image a segmentation ground truth mask is required during training. However, they rely on pixel-wise class labels, and for histopathology images, such annotations must be performed by medical experts. This is highly impractical, as the annotation of giga-pixel WSIs on pixel-level is quite time-consuming. Therefore, weakly-supervised semantic segmentation (WSSS) algorithms are often applied which can cope with weak training labels such as inexact, incomplete, or inaccurate annotations [13]. Here, the annotations used for training contain less spatial information than pixel-wise class labels, making it more challenging to obtain a precise and highly-resolved segmentation. Examples of common training annotations in WSSS algorithms include bounding boxes, scribbles, points, and image labels [13].

[1] Source: Adapted from Figure 1 in [6]

Several WSSS methods have been proposed for histopathological images [13]. Many of them perform segmentation of tumorous tissue based on binary ground truth labels on patch-level [27, 50, 56, 80]. However, most methods produce a relatively low-resolution segmentation (i.e. $> 1\,\mu m$/pixel) or suffer from oversegmentation. On the one hand, this is because they choose to operate on low-resolution WSIs. For another, some models perform WSI segmentation by classifying entire patches as tumorous or not. In this case, the resolution suffers because the patch size is quite large. However, it cannot be chosen arbitrarily small since beyond a certain size, there would not be enough context for the network to determine whether the depicted tissue is tumorous or not. Most WSSS approaches have been studied on low-resolution WSIs and it is unclear if existing methods (when adapted accordingly) can yield a precise segmentation for highly resolved histopathology images (i.e. $\geq 1\,\mu m$/pixel). However, an accurate high-resolution segmentation can be the basis for a reliable and robust downstream analysis. Deep learning models for disease prognosis or classification could learn from the detailed features extracted from high-resolution segmentation to improve their predictive performance. In addition, higher resolution could improve the accuracy of quantifying various tissue components (such as tumor cells, immune cells, stroma), which are often used to diagnose disease and assess response to therapy.

Therefore, this master thesis explores a WSSS approach for high-resolution WSIs and a pretraining strategy to further improve its segmentation performance. The WSI dataset in this thesis contains incomplete and inexact regional annotations of the cancerous disease colorectal adenocarcinoma as well as healthy tissue. The goal was to exploit these weak annotations to perform a highly-resolved semantic segmentation of tumorous tissue on pixel level. This is equivalent to assigning a binary class label to each image pixel.

More precisely, the state-of-the-art model SwinMIL [50] was used to perform the WSSS. SwinMIL is a multiple-instance learning (MIL) based model which trains on image-level labels and provides a pixel-wise segmentation for each input image. Hence, patches extracted from the weakly annotated regions of the WSIs were used to train this model. The pretraining strategy is incremental and aims to improve the performance of SwinMIL. For this purpose, the Swin Transformer [54] backbone of SwinMIL was pretrained using the self-supervised learning (SSL) method DINO [12]. Since DINO does not require any image labels during training, it was trained using WSI patches from unannotated regions. The performance of both models was evaluated and compared with respect to common segmentation metrics which are further described in Chapter 5.

Another aspect that distinguishes this thesis from previous WSSS work is that the employed dataset does not provide any pixel-wise ground truth masks for the evaluation of the semantic segmentation algorithms. Therefore, a new method of generating pixel-wise ground truth masks for tissue affected by adenocarcinoma is proposed and implemented in this thesis. For this, IHC

stains and the weak annotations of the WSIs are utilized. The exact process is explained in Chapter 4. The generated masks will be used to evaluate the final segmentation performance of the WSSS model SwinMIL. Although SwinMIL is trained using image-level labels, the evaluation still relies on these ground truth masks.

1.1 Contributions

To summarize the previous paragraphs and to give an overview, the main contributions are:

(1) **Proposing and implementing a method to generate pixel-wise ground-truth masks for cancerous regions utilizing IHC stains and weak annotations**
In most cases, the generated masks correctly separate the epithelial tissue from the background and other cell types in the IHC stains. However, the algorithm has difficulty excluding artifacts in the IHC stain that have an intense brown color similar to that of epithelial tissue. In addition, it has difficulty identifying the boundaries of epithelial tissue precisely when it is surrounded by many bluish immune cells.

(2) **Performing different experiments to investigate the performance of the state-of-the-art WSSS model SwinMIL on high-resolution WSIs**
The results showed that SwinMIL produced many false positives in *healthy* slides and had particular difficulty distinguishing between cancerous and non-cancerous epithelial tissue. However, this problem was significantly reduced by hard negative mining, where false positive examples were selectively added to the training dataset. In addition, the weighting of the different stages in SwinMIL has been shown to play a key role in balancing high- and low-level information, which significantly affects pixel-level performance. Contrary to expectations, a qualitative comparison with lower resolved segmentation masks showed that a resolution of less than 1 microns per pixel (mpp) is probably sufficient to segment most of the structures in cancerous epithelial tissue precisely.

(3) **Investigating the effect of self-supervised pretraining with DINO on a Swin Transformer backbone and in the context of histopathological image segmentation**
Compared to an ImageNet initialization of the Swin Transformer backbone, a pretraining with DINO on histopathological image data did not improve the segmentation performance. In fact, it even had a negative effect and led to a slower convergence of the SwinMIL model.

1.2 Thesis Structure

This work is structured into eight chapters. To give a better understanding of the context in which semantic segmentation can be applied for histological analysis, Chapter 2 first lays the medical and technical foundations. Starting with the medical background (Section 2.1), this includes the process of cancer diagnosis via biopsy, i.e. the preparation of tissue and employed staining methods, and the criteria used in cancer prognosis. The characteristics of colon cancer and specifically colorectal adenocarcinoma are highlighted and the challenges in computational pathology with emphasis on deep learning algorithms are discussed. This is followed by the theoretical background on neural networks in Section 2.2 and on hyperparameter optimization in Section 2.3. Furthermore, general methods of performing semantic segmentation and WSSS specifically are described in Section 2.4. The latter is then underlined with concrete approaches from related literature in Chapter 3. The following Chapter 4 provides insights on the employed materials and methods, starting with a detailed description of the WSI datasets (Section 4.1) and the preprocessing methods used to prepare them for neural network training (Section 4.2). Next, the baseline model for tumor segmentation is introduced (Section 4.4.2) providing a performance reference for the main contributions of this thesis: These involve the method for the generation of ground truth masks and their usage for pixel-wise evaluation (Section 4.3), as well as the application of the SwinMIL model (Section 3.2.3) on the problem of high-resolution tumor segmentation and the attempt to improve performance with DINO (Section 3.3). The latter includes their functionality and a description of the performed experiments which aimed to increase the segmentation performance. The chapter concludes with the implementation details on the training of the neural network models (Section 4.5). Subsequently, Chapter 5 presents the qualitative and quantitative results of the performed experiments. The discussion in Chapter 6 relates the results to state-of-the-art approaches from the literature and elaborates on the limitations of the methods presented in this thesis. Based on these, Chapter 7 makes an outlook on which directions can be taken in future work to answer the research questions that have emerged from the experiments. Finally, Chapter 8 summarizes and concludes this thesis.

2 Background

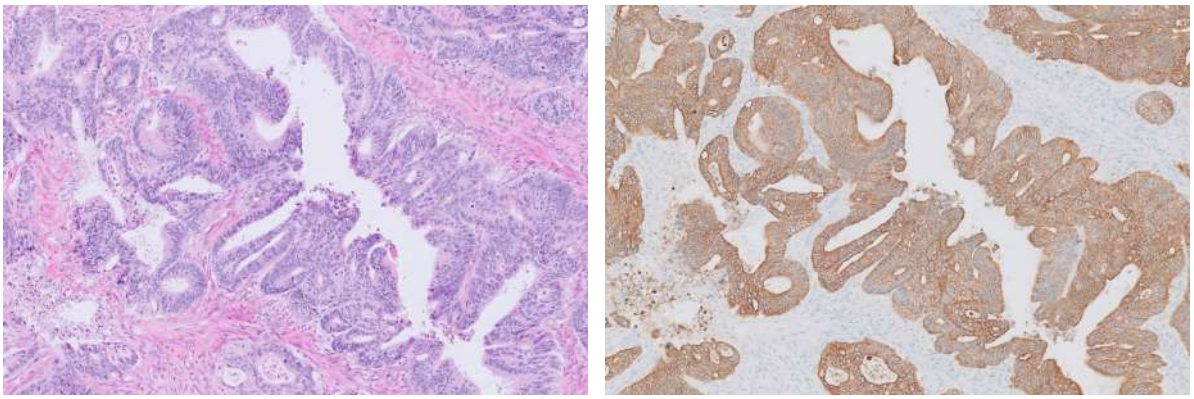
2.1 Histopathology

Histopathology is the branch of pathology that studies the microscopic anatomy of tissues and organs in order to diagnose diseases [62]. It involves the examination of tissue samples, which are usually obtained through biopsies or surgical procedures, to identify any abnormalities in cell structure and organization. Histopathology is an important tool that can provide pathologists with information about the characteristics of cancer such as anaplasia, invasiveness, and tumor heterogeneity, as well as the cancer type and stage [48]. Based on this diagnosis, a prediction of the patient outcome is established and appropriate treatment strategies can be selected [62].

2.1.1 Sample Preparation

After a tissue sample is taken from the patient, several steps must be taken before pathologists can analyze the specimen under a microscope or via digital WSIs [62]. Generally, the removed tissue mass is first prepared into thin slices which are then stained to visualize the different cell and tissue structures under a microscope. Thereby, specific parts of the protocol, such as the employed methods and chemicals, depend on the tissue specimen and the method of observation [70]. The following describes the preparation of so-called formalin-fixed paraffin-embedded (FFPE) samples, a prevalent procedure used to conserve and stabilize biological tissues for light microscopy [62].

The first step begins with the fixation of the removed tissue mass. The main purpose is to prevent the biological deterioration of the sample, i.e. to preserve the morphology and structure of cells, but it also hardens the tissue, preparing it for further processing [74]. One of the most commonly used fixative solutions in sample preparation for light microscopy is formalin [74]. After fixation, the tissue mass is embedded in a more robust medium. This step is essential to increase the mechanical strength and stability of the specimen to allow for easier sectioning into thin slices afterward [70]. For embedding, the tissue sample is placed in a small mold, which is then gradually filled with molten paraffin wax [77]. The wax is then cooled off and solidifies, forming a solid block containing the specimen inside. Prior to embedding, the water is removed from the tissue in a series of dehydration steps using alcohol of increasing concentrations [74]. This is necessary as otherwise, paraffin cannot infiltrate the tissue sample for consolidation since it is insoluble in water. Nowadays, the processes of fixation and embedding are often automated by so-called tissue processors. This is more efficient and has the advantage of eliminating variations in the procedure. In this way, the influence of tissue processing as a source of variation between experimental results on different slides is reduced to a minimum [77]. After embed-



(a) HE Staining (b) IHC Staining

Figure 2.1: Two example images of the same WSI section containing glandular tissue with epithelial cells. The tissue section in (a) was stained with HE. Image (b) shows the result after IHC staining was applied to highlight the protein keratin which is prevalent in epithelial cell walls. The enzymatic reaction of the involved antibodies results in a brown color.

ding, the specimen is finally cut into several tissue sections (typically $3 - 5\,\mu m$ thick) using a microtome and mounted on a glass microscope slide [62].

2.1.2 Staining Techniques

Since thin tissue sections are inherently almost transparent and have low contrast, staining is used to reveal their structures and details [62]. HE staining is one of the most commonly used methods in histology when the goal is to visualize the general structure of the tissue [70]. If specific features or chemical components should be highlighted, IHC staining can be used [70]. For the analysis of tumors, both stains are often used in conjunction, since pathologists may miss tumors with small diameters on the HE stain [57]. In the following, the properties and functionality of both methods will be explained in more detail.

Hematoxylin-Eosin Staining

During HE staining two different colored staining agents, hematoxylin and eosin, are applied sequentially onto the specimen. If a paraffin-embedding was used, the wax is first removed from the tissue section with a solvent and the sample is rehydrated. While there is no standard procedure and laboratory protocols may differ in detail [49], the staining procedures are fundamentally consistent. Figure 2.1a shows a tissue section stained with hematoxylin and eosin. The basic dye hematoxylin stains acidic structures such as cell nuclei dark purple-blue, and the acidic dye eosin stains basic structures such as cytoplasm and components of the extracellular matrix (e.g., collagen and enzymes) in pink-red. Different structures can take on various shades

and combinations of these colors [79]. This allows pathologists to identify cell morphology and organization that are characteristic of certain tumor diseases.

Immunohistochemical Staining

IHC staining is used to detect and localize specific proteins and other antigens in tissue samples [67]. This process involves the use of antibodies that bind to specific target proteins and produce a visible signal, such as a color change or fluorescent light, that can be observed under a microscope. Figure 2.1b shows a tissue section where the protein keratin appears brown after IHC staining was applied. In general, there are two methods to detect the presence of antigens [67]: The direct method uses a single primary antibody that is directly labeled with a detectable marker (i.e. enzyme or fluorescent dye). The primary antibody binds directly to the target antigen and the labeled marker produces a visible signal. Conversely, the indirect method requires two steps: First, an unlabeled primary antibody binds to the target antigen. Then, a second antibody is introduced, which is labeled with a marker and reacts with the primary antibody, allowing for visualization of the antigens. In general, the indirect method is more sensitive compared to the direct method since multiple secondary antibodies can bind to each primary antibody, amplifying the produced signal [68]. However, it also requires an additional step and the use of multiple reagents [68].

2.1.3 Criteria in Cancer Treatment and Prognosis

Once cancer is diagnosed, a decision on the treatment regimen is made and in some cases, a prognosis on the course and outcome of the disease can be established. Treatment plans are often selected in a collaborative process of multiple healthcare specialists (e.g. oncologists, surgeons) and the patient [16]. In general, they are individualized based on the characteristics of each patient and their cancer. Important factors include the age and overall health of the patient, as well as the type, stage, and location of cancer. These factors not only affect the treatment but also the course and outcome of the disease [36]. The next two paragraphs will address the classification of cancer based on grading and staging.

Tumor Grading

The tumor grade refers to the level of abnormality of cancer cells and is determined by examining tissue samples under a microscope [39]. Thereby, pathologists compare tumor cells to normal cells of the same tissue type to assess their abnormality. In general, the more abnormal the cells look, the more aggressive the cancer and the higher its grade [39]. Tumors with higher grade grow faster and spread more rapidly to other parts of the body. Figure 1.1 gives an impression of the visual difference between low and high-grade cancer on a microscopic level.

Low-grade cancer cells are often well differentiated, i.e. they look similar to surrounding normal cells [72]. Higher-grade cancer tends to be poorly differentiated (anaplastic), i.e. normal structures and tissue patterns are missing. The system for tumor grading may depend on the type of cancer, but most tumors can be assigned one of the following five grades [62]:

- Grade X: Grade cannot be accessed
- Grade 1: Well-differentiated
- Grade 2: Moderately differentiated
- Grade 3: Poorly differentiated tumors
- Grade 4: Undifferentiated tumors

Some systems have less than five grades. Often more aggressive treatment (e.g. chemotherapy or radiation therapy) is required to treat higher-graded cancer.

Tumor Staging

The cancer stage is used to describe the extent or spread of cancer within the body [38]. It is typically based on several factors, including the size of the primary tumor, whether it has spread to nearby lymph nodes, and whether it has metastasized to other parts of the body. Although there exist specific staging systems for certain cancer types (e.g. for blood or brain cancer), the most widely used staging system that considers all of the mentioned factors is the TNM system. It refers to the three factors **T**umor, **N**ode, and **M**etastasis, and assigns a number to each of them, with higher numbers indicating a more advanced stage of cancer [72]:

- **Primary Tumor** (T0 – T4)
  This stage describes the size of the primary tumor and whether it has invaded nearby tissues. T0 indicates that there is no evident tumor, while T4 indicates a large tumor that has grown far into nearby tissues.
- **Regional Lymph Nodes** (N0 – N3)
  This stage describes whether the cancer has spread to nearby lymph nodes. N0 means that there is no spread to lymph nodes, while N3 means that the cancer has spread to multiple lymph nodes.
- **Metastasis** (M0 or M1)
  This stage refers to whether the cancer has spread to other parts of the body. M0 indicates that there is no evidence of metastasis, while M1 indicates that the cancer has spread to distant organs or tissues.

For some cancer types, the system is extended by more detailed subclassifications (e.g. T3a and T3b) [72]. Additionally, prefixes can be used to express how the stage was determined. For

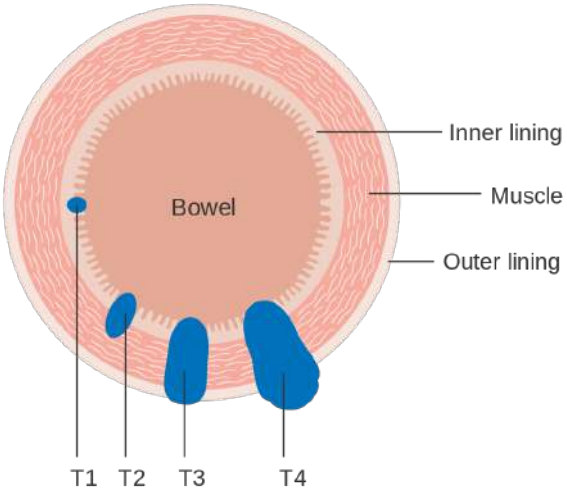


Figure 2.2: Different T stages of bowel cancer. The tumor grows outwards starting at the inner lining and infiltrates the different layers of the intestinal wall. Higher tumor stages (T3 or T4) can even penetrate the outer lining of the bowel wall.[2]

example, pT1 indicates a low-stage tumor which was determined by pathological examination of a tissue sample.

2.1.4 Colorectal Cancer

Colorectal Cancer is the third most common type of cancer accounting for approximately 10% of all cancer-related deaths worldwide [78]. It affects the colon (large intestine) and the rectum, which are part of the digestive system. The symptoms depend on the location of the tumor in the bowel and around 50% of the people do not experience any symptoms at all [42]. Common symptoms include changes in bowel habits (e.g. diarrhea or constipation), blood in stool, and loss of appetite and weight [71]. Colorectal cancer can be classified into more specific histopathological types such as adenocarcinoma, lymphoma, or carcinoid tumors, which mainly differ in the type of cells that have become cancerous. The deep learning algorithms in this thesis were developed on WSIs of colorectal adenocarcinoma.

Colorectal Adenocarcinoma

Adenocarcinoma is the most prevalent type of colorectal cancer and accounts for approximately 98% of all cases [37]. It arises from glandular epithelial cells which are arranged into structures called glands and line the inner surface of organs such as the colon or rectum [70]. These cells produce and secrete substances like digestive enzymes, hormones, and mucus [70]. In colorectal adenocarcinoma, a malignant epithelial tumor usually starts growing from the inner

[2] Source: Figure 3 from [75]

lining of the bowel and continues growing outwards penetrating the bowel wall. Thereby, larger tumors that have invaded further into the bowel wall are classified in a higher T stage according to the TNM system (see Figure 2.2).

2.1.5 Digital Pathology

Digital pathology is the practice of converting glass slides containing tissue samples into digital WSIs that can be viewed, analyzed, and managed on a computer [74]. This involves scanning the glass slides at high resolution and storing the resulting images in a digital format. As mentioned in the introduction (Chapter 1), it has many advantages over traditional microscopy, including more efficient and accurate diagnoses [45], as well as enhanced collaboration between pathologists [22].

Whole Slide Images

WSIs are high-resolution digital images of entire glass slides and are acquired using specialized scanners which capture entire tissue sections at high magnification. Depending on the magnification level, the resulting WSIs can have sizes in the order of $100,000 \times 100,000$ pixels and consume a considerable amount of disk space with each image file ranging from hundreds of megabytes to several gigabytes [5]. Typically, WSIs are stored as compressed TIFF images organized into a hierarchy of different resolutions [35]. This pyramid of images is created by downsampling the original high-resolution image into a series of smaller images, each representing a different level of magnification. The highest magnification used for machine learning is often 20× [10] which refers to a resolution of 0.5 mpp by convention [35]. For every subsequent level, the WSIs is usually downscaled by a factor of 2, i.e. a WSI at level n represents the original image downsampled by a factor of 2^n [35]. Thereby, different resolution levels expose different features and structures: On lower resolutions around $5-10$ mpp, the entire slide can be visualized quite well, but individual cells or structures cannot be identified. However, this level is useful for navigating and orienting oneself on the slide and identifying regions of interest. On an intermediate-resolution resolution level around $0.5-2$ mpp individual cells and structures become visible. This level is useful for identifying tissue types, detecting gross morphological changes, and evaluating tissue architecture. Finally, on a high-resolution level of $0.1-0.5$ mpp subcellular structures, and individual nuclei are revealed, which is useful for identifying and characterizing specific cells or morphological changes at the cellular level. Thus, depending on the specific task pathologists or image analysis algorithms may use different WSI resolutions.

Computational Pathology

Computational pathology is a subfield of digital pathology that focuses on the application of computer algorithms to the analysis of pathological images such as WSIs. The main goal is to

extract meaningful information that can support pathologists to make a more accurate and efficient diagnosis or prognosis. For example, an automatic segmentation or detection of specific objects such as tumor cells or cell nuclei can be performed. A delineation of object boundaries can help to determine relevant quantitative metrics such as the number of objects, their individual size, and the area of certain structures [19]. For this purpose, deep learning algorithms have demonstrated superior performance compared to traditional computer vision algorithms. The main reason is their ability to automatically extract complex features from large, high-dimensional datasets [45]. Traditional computer vision methods often require manual feature engineering, which can be time-consuming and difficult in the context of WSIs.

Although deep learning has become the prevalent method in computational pathology, several challenges have to be addressed to ensure its successful adoption on WSIs. One is the high dimensionality of WSIs [45]. Neural networks generally require more memory and computing power the larger the images they process. This makes it infeasible to process giga-pixel WSIs in high resolution as a whole. Therefore, WSIs are often divided into smaller patches where each patch is then processed individually. To obtain a prediction on WSI level, the predictions of the patches can then be aggregated. In the case of object recognition and segmentation problems, the predictions of each patch can be stitched together to create the final prediction mask for the entire WSI. However, the problem with tiling into smaller patches is that context information can get lost. For example, tumor cells may not be identified correctly because the higher-level tissue structure is not visible in the patch.

Another challenge is the generalization of deep learning algorithms to new datasets [46]. It is desirable to have a robust model that also performs well when conditions vary to a certain extent. In the case of WSIs, one obstacle is that datasets are often not sufficiently large and have little variability, i.e. are not representative [45]. The reasons for datasets with small sample sizes are the high annotation costs of WSIs and the often restricted access to medical data. A further factor that may impede the generalization of deep learning models is the stain variability between different laboratories. WSIs can exhibit significant variations in color, contrast, and illumination between different datasets. This can be caused by differences in the tissue preparation and staining protocol, as well as by different characteristics of employed whole-slide scanners [45]. Stain variations and insufficient training samples can decrease the performance of deep learning models. Good ways to improve the generalization of neural networks include data augmentation to produce more training samples and reflect the variability in the data distribution, as well as color normalization.

2.2 Deep Neural Networks

Machine learning has emerged as a powerful tool in the field of computer vision, enabling the development of algorithms that can automatically analyze and interpret visual data. Especially, the use of deep neural networks (DNNs) has gained significant attention across multiple domains in recent years. One of the main reasons for their success is their ability to extract high-level features from huge datasets. For a long time, feature extractors were designed manually which is time-consuming, requires extensive knowledge in the corresponding application domain, and carries the risk of not capturing all relevant information in the data. In contrast, DNNs can automatically generate suitable data representations known as feature vectors from raw data such as the pixel values of images. These feature vectors can then be used to detect and classify certain patterns in the input. Since this thesis aims to explore the current state-of-the-art techniques of DNNs for WSSS on histopathological image data, the following sections will introduce the main concepts employed in DNNs for image processing.

2.2.1 Fundamental Concepts

In image processing, neural networks can be viewed as an architecture that computes a non-linear mapping $f(\boldsymbol{x})$ between a raw input image $\boldsymbol{x}$ and a task-dependent output (e.g. a segmentation map). For this, several modules are connected in series, each transforming the input into a higher, slightly more abstract representation. By stacking multiple of these non-linear modules, or layers, a neural network can learn a complex high-dimensional function that amplifies the important features of the input image and suppresses irrelevant information. Each layer in a neural network can also be seen as a simple, potentially non-linear, function that often depends upon adjustable parameters $\boldsymbol{\theta} = \begin{pmatrix} \boldsymbol{W} & \boldsymbol{b} \end{pmatrix}$, known as weights and biases. The key aspect of neural networks is that these internal parameters $\boldsymbol{\theta}$ are learned from data, a process referred to as training. Therefore, input images are passed through the network to generate an output (forward pass), followed by an adjustment of the parameters $\boldsymbol{\theta}$ to approximate the target output (backward pass). This process is usually repeated until the neural network has converged and learned the desired mapping $f(\boldsymbol{x})$. To guide this learning process, a so-called objective function $J(\boldsymbol{\theta})$ is introduced as the final layer of the network during training. It measures the error, which is the distance between the desired output and the actual output. During the training process, the internal parameters $\boldsymbol{\theta}$ of the network are then adapted to minimize $J(\boldsymbol{\theta})$ using the concept of backpropagation and gradient descent. The following sections describe the parameter optimization in more detail.

Objective Function

The objective function is typically defined as an average over the samples of the training set. In supervised learning, where each training sample x_i has a corresponding ground truth label y_i, the objective function can be written as:

$$J(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} L\left(f\left(x_i; \boldsymbol{\theta}\right), y_i\right) \tag{2.1}$$

Here, the average is computed over m training samples whereby L is the per-sample loss function which depends on the network output $f\left(x_i; \boldsymbol{\theta}\right)$ and the target output y_i for a training sample x_i. The objective function can be seen as a noisy landscape in a high-dimensional space, where each point on the surface corresponds to a particular set of parameter values. Depending on the particular task and the properties of the dataset, the loss function may depend upon different parameters. For example, $\boldsymbol{x}$ and $\boldsymbol{\theta}$ can be added to the arguments of the loss function if it should include a form of regularization, or the ground truth labels $\boldsymbol{y}$ can be omitted to formulate an unsupervised learning problem. A commonly used loss function for multi-class segmentation and classification problems is the cross-entropy loss. It interprets the outputs of the model and their corresponding ground-truth labels as two distinct probability distributions and measures the difference between them. The cross-entropy loss is given by:

$$L\left(\hat{\boldsymbol{y}}, \boldsymbol{y}\right) = -\sum_{i=1}^{K} y_i \log\left(\hat{y}_i\right) \tag{2.2}$$

where $\hat{\boldsymbol{y}}$ is the predicted probability vector and $\boldsymbol{y}$ is the corresponding label vector. Both vectors have a length of K, where K is the number of classes. The entries $\hat{y}_i$ of the predicted vector indicate the probability for class i. The label vector is one-hot encoded and entries where $y_i = 1$ indicate the corresponding class.

Training Neural Networks

When training a neural network from scratch, the process starts with a random initialization of the internal parameters $\boldsymbol{\theta}$. The parameters are initialized with values drawn from a certain probability distribution, such as a uniform distribution. However, most of the time, more sophisticated initialization methods such as He [28] or Xavier [24] initialization are used because they generally lead to higher accuracy and faster convergence.

To compute the gradient of the average loss in equation 2.3 with stochastic gradient descent (SGD), a procedure called backpropagation is applied. The basic idea of backpropagation is to calculate the error at the output layer, and then propagate this error backward through the network to compute the gradients of the objective function with respect to the parameters of each layer. These local gradients are then used by SGD to update the parameters of each layer.

Backpropagation is essentially the application of the chain rule for calculating the derivative of a nested function. In this case, the chain rule is applied to the objective function and each layer constitutes a further inner function.

To find the optimal parameters $\boldsymbol{\theta}^*$ which minimize the loss function, most neural networks employ a procedure called SGD. First, the output and average loss are computed for a mini-batch of m training samples according to equation 2.1. From this, an approximation of the average gradient $\nabla J(\boldsymbol{\theta}_t)$ with respect to $\boldsymbol{\theta}_t$ is derived:

$$\nabla J(\boldsymbol{\theta}_t) = \frac{1}{m} \nabla_{\boldsymbol{\theta}_t} \sum_{i=1}^{m} L\left(f\left(x_i; \boldsymbol{\theta}_t\right), y_i\right) \tag{2.3}$$

The parameters $\boldsymbol{\theta}_t$ at iteration t are then updated in the direction of the negative gradient estimate which indicates the steepest descent in the landscape of the objective function:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \nabla J(\boldsymbol{\theta}_t) \tag{2.4}$$

The learning rate η is determined empirically and decides the step size in each iteration, that is how much the model parameters should be adjusted in response to the average loss. The parameter update is usually repeated with different subsets (mini-batches) of the training data until the objective function stops decreasing. This process is called stochastic because it only gives a noisy estimate of the gradient of the objective function since the subset is selected randomly from the training set in each iteration. In contrast, batch gradient descent algorithms use the entire training dataset to compute the gradient at each step.

2.2.2 Multilayer Perceptrons

DNNs often employ multiple building blocks with various layer types, each having its own specific purpose. To go from one layer to the next, typically some form of (trainable) aggregation is applied to the input. A common approach to aggregate the output from the previous layer is multilayer perceptrons (MLPs). They consist of multiple fully connected layers, each followed by a non-linear activation function. Although it is possible to build a DNNs architecture for vision using only MLPs, most of the time other components such as convolutional layers (see Section 2.2.3) or attention layers (see Section 2.2.4) are involved as well. The reasons for this are that MLPs are often computationally intensive and prone to overfitting when applied to high-dimensional image data.

Fully-Connected Layers

Fully connected layers perform a linear projection of an input vector $\boldsymbol{x} \in \mathbb{R}^m$ which can be formulated in terms of a matrix-vector multiplication:

$$\boldsymbol{z} = \boldsymbol{W}^\top \boldsymbol{x} + \boldsymbol{b} \tag{2.5}$$

where $\boldsymbol{W} \in \mathbb{R}^{n \times m}$ is the weight matrix whose i-th row is the weight vector $\boldsymbol{w}_i$, and $\boldsymbol{b}$ is a vector of biases given by $\boldsymbol{b} = (b_1, b_2, \ldots, b_n)$. Considering the entries of vector $\boldsymbol{x}$ as input neurons and the entries of vector $\boldsymbol{z}$ as output neurons, a fully connected layer computes a weighted sum of all input neurons for each output neuron. Thus, each output neuron with index i is associated with its own weights $\boldsymbol{w}_i \in \mathbb{R}^m$ and bias $b_i \in \mathbb{R}$, i.e. the parameters are not shared.

Activation Functions

Activation functions are essential for introducing non-linearity into the model, allowing the network to learn complex, non-linear relationships between the input data and output. Using only linear modules such as fully-connected layers without activation functions would limit the model's ability to represent complex relationships. An activation function h is applied element-wise to the output of the previous layer $\boldsymbol{z}$:

$$\hat{\boldsymbol{y}} = h(\boldsymbol{z}) \tag{2.6}$$

In the case of MLPs, $\boldsymbol{z}$ is the linear transformation in equation 2.5. A common choice for the activation function in DNNs is the rectified linear unit (ReLU), since it helps to avoid the vanishing gradient problem (see Section 2.2.3). It is a piecewise linear function defined as:

$$h(x) = \mathrm{ReLU}(x) = \begin{cases} x, & \text{if } x > 0 \\ 0, & \text{else} \end{cases} \tag{2.7}$$

Since the derivative of the ReLU function for $x > 0$ equals 1 the gradient is passed through unchanged during backpropagation for positive values of x. This property helps to maintain the gradient signal throughout backpropagation up to the earliest layers of the network, avoiding the vanishing gradient problem. However, an issue that can occur in neural networks using ReLU activation functions is the "dying ReLU" problem. This happens when the input neurons to the ReLU functions are always negative, in which case ReLU evaluates to zero. Once in this state, it does not participate in discriminating the inputs anymore. It cannot recover either, since the backpropagated gradient for negative inputs is also zero and provides no signal for the weight update of the previous neuron to change ReLU's input.

2.2.3 Convolutional Neural Networks

Convolutional neural networks (CNNs) are a type of artificial neural network that has gained immense popularity in the field of computer vision over the past decade. They are specifically designed to process multidimensional data like images or videos and achieve state-of-the-art performance on a wide range of visual recognition tasks, such as image classification, object detection, and semantic segmentation. A typical CNN processes data in a hierarchical way by applying multiple stages of convolutional and pooling layers. Early layers recognize simple

features such as corners and edges, which are combined into more complex and abstract features in deeper layers. This hierarchical approach is inspired by the characteristics of natural signals, where higher-level features are often composed of lower-level ones. For example, in images, an object like a car is composed of tires, doors, windows, and several other components. The three main characteristics of CNNs, which also distinguish them from MLPs, are local connections, shared weights, and pooling. The following two paragraphs describe the two core components of CNNs: convolutional and pooling layers.

Convolutional Layers

A convolutional layer is designed to detect local patterns or features in the input data by convolving a set of filters over the input. It usually takes an input tensor of shape (C_{in}, H_{in}, W_{in}), where C_{in} is the number of input channels (e.g. 3 for RGB images), and H_{in} and W_{in} are the height and width of the input. The input tensor is convolved by a set of learnable filters (kernels), each of shape (C_{in}, k_h, k_w), where k_h and k_w are the height and width of the filter. Thereby, each convolution with a specific kernel yields a corresponding two-dimensional output referred to as feature map. Descriptively, a convolution involves sliding each kernel over the input tensor one unit at a time. At each position, the values where the kernel and input tensor overlap are multiplied element-wise, and the resulting values are summed up to produce a single output value. Essentially, a convolution computes a locally weighted sum over its input. Mathematically, this can be expressed as:

$$y_{i,j} = \sum_{c=1}^{C_{in}} \sum_{p=1}^{k_h} \sum_{q=1}^{k_w} x_{c,i+p-1,j+q-1} \cdot w_{c,p,q} \tag{2.8}$$

where $y_{i,j}$ is the value at position (i, j) of the two-dimensional feature map, $x_{c,i+p-1,j+q-1}$ is the input at channel c and position $(i + p - 1, j + q - 1)$, and $w_{c,p,q}$ is the weight of the kernel at channel c and position (p, q). The values $y_{i,j}$ are then passed through a non-linear activation function.

In contrast to MLPs, convolution has the following two properties: It operates locally and the output neurons share their weights (within one feature map). The former property contributes to better recognition of local features. In images, neighboring pixels are often highly correlated and form motifs that local filters can easily recognize without being affected by the context of the whole image. The second property allows for the detection of motifs in the entire image regardless of their position, because the same filter is applied to all different locations in the image.

Pooling Layers

Pooling layers are the second crucial building block of CNNs and typically follow after convolutional layers. They reduce the spatial size of the input by summarizing neighboring values

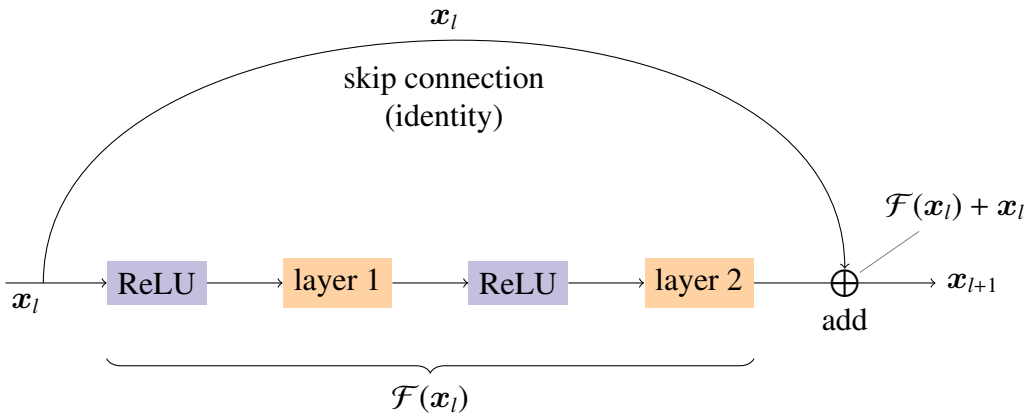


Figure 2.3: Example residual block. The residual function $\mathcal{F}$ can be represented by different numbers and types of layers. For example, layers 1 and 2 can be implemented using batch normalization followed by convolution [29].

and at the same time preserve the important features. This allows the network to reduce the number of parameters which leads to less computational cost and also helps to prevent overfitting. Pooling layers also follow the sliding window approach, but unlike standard convolutions, they operate only on the spatial dimensions (not the channel dimension) of the input tensor, i.e. each feature map is downsampled independently. There are several types of pooling layers but the most common one is max pooling, where the maximum value within a certain window size is taken. Pooling also plays an important role in creating an invariance to small translations and distortions of features.

Residual Connections

A well-known challenge in DNNs is the vanishing gradient, which occurs when the gradient signal becomes too small as it propagates through multiple layers of the network during the backward pass. This impedes the learning process of the earlier layers, as their weights are updated very slowly, if at all. The vanishing gradient problem arises because the gradient of the loss with respect to the weights is computed recursively using the chain rule of calculus, which involves multiplying multiple derivatives together. As a result, the gradients will only have a minimal impact on the weights when updated according to equation 2.4.

To address this problem, a number of techniques have been developed. One of them is residual connections, which were originally proposed in residual neural networks (ResNets) [29] and became a standard component of many state-of-the-art DNNs architectures by now. Residual connections (also known as skip connections) allow for direct information flow from earlier to later layers in a neural network. This is often implemented using so-called residual blocks

where the output of a block is added to its input before the result is passed to the next layer. A residual block is visualized in Figure 2.3 and can be expressed by the formula:

$$\boldsymbol{x}_{l+1} = \boldsymbol{x}_l + \mathcal{F}(\boldsymbol{x}_l, \boldsymbol{W}_l) \tag{2.9}$$

where $\boldsymbol{x}_l$ and $\boldsymbol{x}_{l+1}$ are the input and output of the l-th residual block, and $\mathcal{F}$ represents the transformation performed by the residual block. The specific layer types used to implement $\mathcal{F}$ can vary between different architectures. The idea of residual connections is to learn the residual function $\mathcal{F}$ which computes the difference between input and the desired output (rearranging equation 2.9: $\mathcal{F}(\boldsymbol{x}_l, \mathcal{W}_l) = \boldsymbol{x}_{l+1} - \boldsymbol{x}_l$), rather than trying to learn the output $\boldsymbol{x}_{l+1}$ directly. This approach has been shown effective in preventing vanishing gradients, enabling the training of DNNs with hundreds of layers.

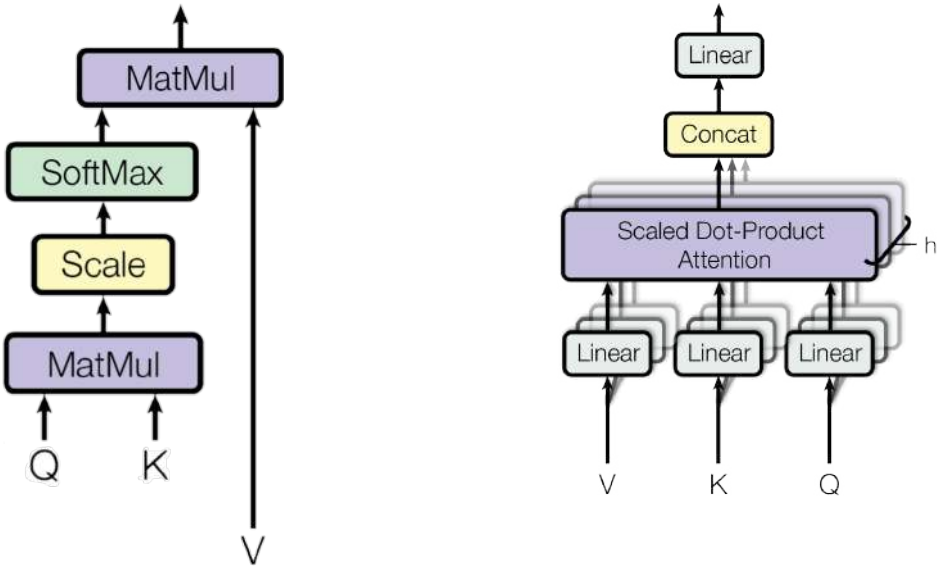
2.2.4 Transformer Architecture for Vision

The Transformer architecture was initially proposed in 2017 by Vaswani et al. [76] for natural language processing (NLP) tasks and has become the dominant approach to building large language models since then. However, in recent years, this architecture has also gained huge popularity in computer vision, as several works focused on adapting its concept of so-called self-attention for image processing tasks. Nowadays, self-attention is an essential component in many state-of-the-art DNNs for image classification, segmentation, or object detection. The most significant cornerstone for this development was probably laid by the work of Dosovitskiy et al. [20] in 2020. It introduced the Vision Transformer model which entirely discards convolutions and relies on the self-attention mechanism instead. Although there had already been prior attempts to integrate self-attention into CNNs, and even completely avoiding convolutions, the Vision Transformer was the first non-convolutional model which could be effectively scaled on modern hardware accelerators such as GPUs. The next two paragraphs discuss the general principle of attention in deep learning and its implementation in the Vision Transformer architecture.

Attention Mechanism

The attention mechanism operates on input sequences and allows the model to selectively focus on different parts of that sequence to learn meaningful representations of the data. Single elements from this input sequence are referred to as tokens. A token is a vector embedding of a particular entity that depends on the specific task and domain. In the context of computer vision, a token typically represents the embedding of either an entire image or parts of it (e.g. patches).

The core component of the attention mechanism is the attention function. It takes a query token and a sequence of key-value token pairs as input and can be described as a mapping that relates



(a) Scaled dot-product attention **(b)** Multi-head attention

Figure 2.4: A common implementation of the attention mechanism involves the scaled dot-product attention depicted in (a). It first calculates the attention values via dot-product between queries $\boldsymbol{Q}$ and keys $\boldsymbol{K}$, and uses these values to compute a weighted sum of all values $\boldsymbol{V}$. In practice, it showed beneficial to use multiple heads, i.e. to linearly project each sequence h times and apply attention to all subspaces in parallel as shown in (b).[3]

the query to each value by computing a set of attention weights. The attention weights can be calculated in different ways, but one of the most common ways is to take the scaled dot-product between the query vector and each key vector and apply the softmax function. Loosely speaking, the attention weights indicate the importance of each value token for the query, or to put it another way, how much the query attends to each token of the value sequence. The final output of the attention function is computed by taking the weighted sum of all value tokens. In practice, the scaled dot-product attention is computed on n queries simultaneously, using a sequence of m key-value pairs. This is depicted in Figure 2.4a and can be expressed by the formula:

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}}\right)\boldsymbol{V} \tag{2.10}$$

where $\boldsymbol{Q} \in \mathbb{R}^{n \times d_k}$ is a matrix of all queries, and $\boldsymbol{K} \in \mathbb{R}^{m \times d_k}$ and $\boldsymbol{V} \in \mathbb{R}^{m \times d_v}$ are matrices containing the keys and values, respectively. The dimension of each query and key vector is d_k and the dimension of the value vectors is d_v. The resulting matrix with dimensions $n \times d_v$ contains an output vector for each query. In case $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ represent the same sequence (i.e. $n = m$), equation 2.10 is also referred to as self-attention. For large vector dimensions d_k the dot-products between queries and keys grow large in magnitude, which according to the

[3] Source: Adapted from Figure 2 in [76]

authors Vaswani et al., might push the softmax function into regions with very small gradients. To avoid this, the dot-products are therefore scaled by $\sqrt{\frac{1}{d_k}}$.

Neural networks using the attention mechanism usually apply the attention function multiple times to different representations of queries, keys, and values. The tokens are first projected h times into different subspaces using h learned linear projections. Afterwards, the attention function is applied to each of these projected versions of queries, keys, and values in parallel, yielding n vectors with dimensions d_v for each subspace. Finally, the output vectors from each attention function are concatenated and projected once again, resulting in the final output of n vectors with dimensions d_v. This procedure is referred to as multi-head attention and depicted in Figure 2.4b. It allows the model to jointly attend to different positions in the value sequence at different representation subspaces. Formally, multi-head attention can be expressed as:

$$\begin{aligned} \text{MultiHead}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) &= \text{Concat}\left(\text{head}_1, \ldots, \text{head}_\text{h}\right) \boldsymbol{W}^O \\ \text{where } \text{head}_\text{i} &= \text{Attention}\left(\boldsymbol{Q}\boldsymbol{W}_i^Q, \boldsymbol{K}\boldsymbol{W}_i^K, \boldsymbol{V}\boldsymbol{W}_i^V\right) \end{aligned} \tag{2.11}$$

where each of the h projections is referred to as attention head. The weight matrices $\boldsymbol{W}_i^Q \in \mathbb{R}^{d_{hidden} \times d_k}$, $\boldsymbol{W}_i^K \in \mathbb{R}^{d_{hidden} \times d_k}$, and $\boldsymbol{W}_i^V \in \mathbb{R}^{d_{hidden} \times d_v}$ represent the learned linear projections of attention head i. The final linear projection $\boldsymbol{W}^O \in \mathbb{R}^{hd_v \times d_{hidden}}$ applied after concatenation of all heads is also learned. The dimension d_{hidden} of the latent vectors and the number of heads h are user-defined parameters.

Vision Transformer

The original Vision Transformer is designed for image classification. It learns the representation of an image by dividing it into smaller patches and processing this sequence of patches using the mechanism of multi-head self-attention. Figure 2.5a shows an overview of the model. The Vision Transformer takes an input image $\boldsymbol{x} \in \mathbb{R}^{H \times W \times C}$ where (H, W) are the height and width, and C is the number of channels (e.g. $C = 3$ for RGB images). First, the image is divided into a sequence of quadratic, non-overlapping patches $\boldsymbol{x}_{qp} \in \mathbb{R}^{N \times P \times P \times C}$ where $N = HW/P^2$ is the number of patches and $P \times P$ is the resolution of each patch. The patches of the sequence $\boldsymbol{x}_{qp}$ are then flattened, resulting in a new sequence $\boldsymbol{x}_p \in \mathbb{R}^{N \times (P^2 \cdot C)}$, where every patch is represented by a vector of size $P^2 \cdot C$.

Afterwards, each patch vector is linearly projected into a hidden subspace, resulting in a sequence of N vectors with dimension d_{hidden}. The dimension of these token vectors stays the same throughout the network. Furthermore, a learnable positional embedding is added to each token to inform the model about the relative spatial position of every patch. This is important because unlike CNNs, Transformers do not have an inherent understanding of spatial relationships between different parts of the image. In the next step, an additional $[class]$ token is prepended to the sequence of embedded patches. This neutral token attends to all other tokens

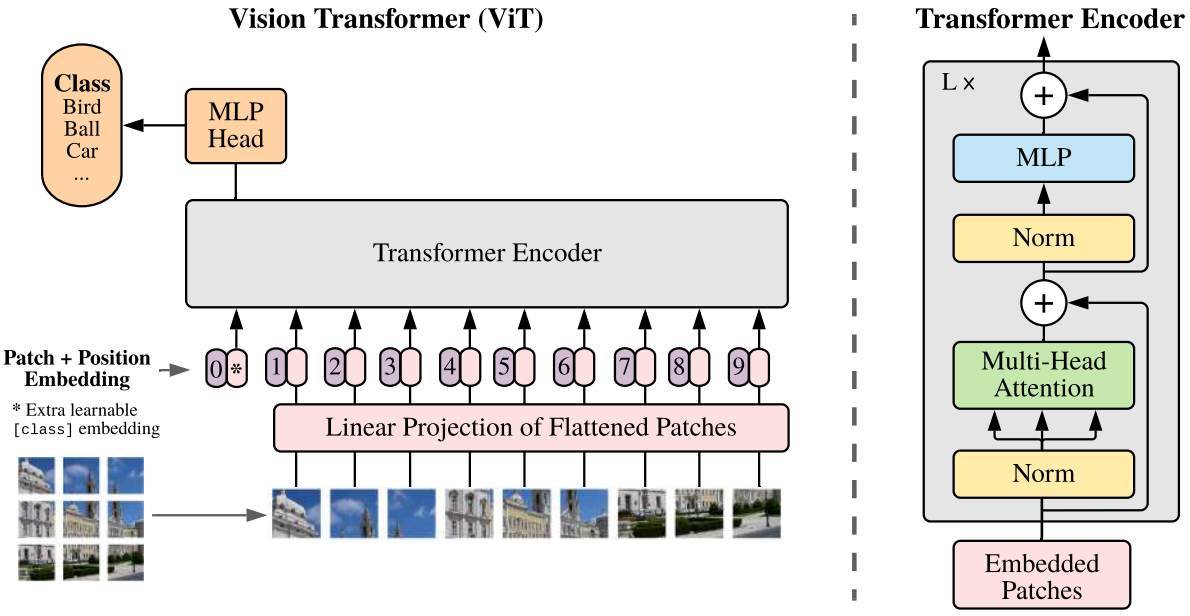


Figure 2.5: The Vision Transformer in (a) divides images into fixed-sized patches and flattens them into a one-dimensional sequence of vectors. These tokens are passed to the transformer encoder in (b) which applies a series of attention and MLP layers. The resulting representations are fed into an MLP classifier to produce the final class output.[4]

from the input sequence and serves as the image representation. Its final state at the output of the model serves as input for the classifier. The sequence of token embeddings is then passed through the Transformer encoder, which consists of multiple identical Transformer blocks (Figure 2.5b). Each block contains a layer of multi-head self-attention followed by an MLP. A layer normalization is applied before both of these components and two skip connections are inserted behind each component. The self-attention mechanism allows the model to attend to different patches in the input sequence, while the MLP applies a non-linear transformation to the output of the self-attention mechanism.

The final output of the Transformer encoder is a sequence of $N + 1$ feature vectors, one for each patch in the image plus the additional [*class*] token. To obtain a single prediction for the entire image, the [*class*] is passed through an MLP classifier head, which outputs the probability distribution over the possible classes.

Although the original Vision Transformer is only capable of image classification, later works have adapted the architecture successfully to perform other vision tasks like semantic segmentation and object detection [8, 73]. Nowadays, the Transformer builds the foundation of many state-of-the-art models for computer vision.

[4] Source: Figure 1 in [20]

2.2.5 Comparison of CNNs and Transformer-based Architectures

CNNs and Transformer-based networks differ in their approach to processing and interpreting visual input data. A concept that can be used to describe the inherent properties of neural networks is the inductive bias. It refers to the set of assumptions that a network uses to learn from data which also influences the model design. The inductive bias can be thought of as a set of restrictions on the types of functions that the neural network can learn. These restrictions can help to guide the learning process by focusing the network's attention on specific aspects of the input data and encouraging it to learn certain types of relationships. Incorporating the right inductive biases into the model for a given task can lead to more accurate results.

CNNs have several inductive biases that are built into their architecture. The first is local connectivity, meaning that each neuron processes information only from a small region of the previous feature map. This means CNNs put more importance on pixels that are in proximity of each other in the early layers. The field of view of CNNs can only grow as the depth of the network increases. In contrast, Transformer-based architectures have a more global perspective of the input image since they capture long-range dependencies already in the early stages of the network. With the attention mechanism, the network can attend to all patches from the input image from the first layer on, rather than being limited to local regions.

The second inductive bias of CNNs is their hierarchical network structure. Convolutional and pooling layers summarize local regions by sliding filters over feature maps producing a scalar value for each region. This procedure results in a pyramid structure where the spatial dimensions of feature maps decrease and the channel dimensions increase throughout the network, allowing to capture features of increasing scales. This reflects the characteristics of natural images, where higher-level features are often obtained by composing lower-level ones. On the other side, Transformer-based networks have an isotropic network design, meaning that the dimensions of the feature maps stay the same. The Vision Transformer projects each image patch into a hidden space producing token vectors of size d_{hidden}. Although the tokens are transformed in different ways throughout the network, their number and shape always stay the same.

A property related to the inductive biases is the receptive field of a neural network. It can be defined for every neuron in the network and tells which information a neuron does perceive. In other words, the receptive field is the size of the region in the input that produces a single output feature. Choosing the right size for the receptive field is important. If it is too small, the network may not be able to encode the relative positions between different features or capture the whole context of objects. The latter is especially important for semantic segmentation or object detection tasks. On the other side, if the receptive field is chosen too large, the network may start to capture global features at the expense of local details. Therefore, it is important to balance the size of the receptive field. While in CNNs the size of the receptive field strictly

depends on the kernel size and the stride chosen for convolution, it is typically of more dynamic nature in Transformer-based architectures. The self-attention mechanism allows Transformer models to attend to all patch embeddings from the first layer on. However, the computed attention weights decide which tokens are integrated to which amount in the new representation. Therefore, the effective receptive field can vary between different stages of the Transformer encoder and depends on the input data.

Overall, Transformer-based architectures have a lower inductive bias and typically a larger receptive field than CNNs. However, because of the low inductive bias, they often require more training data to learn the correct relationship between input and output.

2.3 Hyperparameter Optimization

Hyperparameters are the parameters of a machine learning model which are set by the user. They can have a significant impact on the performance of a machine learning algorithm, and their choice often requires careful consideration. Some examples of hyperparameters for neural networks include learning rate, batch size, regularization parameters, and the number of layers. The choice of hyperparameters is often made through a process of trial and error, by systematically testing different combinations of values and evaluating the performance of the model on a validation set. This is referred to as hyperparameter optimization. This search can be done manually or through automated methods, such as grid search, random search, or more advanced techniques like Bayesian optimization.

In general, there are several important properties that determine the effectiveness of a hyperparameter optimization algorithm. One of them is the balance between exploration and exploitation of the search space. The former refers to exploring a wide range of parameters while the latter is about exploiting promising parameter configurations inferred from previous results. Since the time available for hyperparameter optimization is limited in most scenarios, there is a trade-off between exploration and exploitation. A good hyperparameter optimization algorithm should find the right balance for the given scenario.

Another important factor is the scalability of the optimization algorithm, i.e. its ability to handle large datasets and models efficiently. It would be desirable to perform a search through a large space of hyperparameters without requiring excessive time and computational resources. Especially deep learning models can require days or sometimes even weeks to train and may have many hyperparameters. In such cases, the hyperparameter optimization algorithm should be able to deal with large search spaces and have a good strategy to work with a limited time budget.

The following paragraphs describe three state-of-the-art algorithms for hyperparameter search in more detail, including their strengths and limitations.

2.3.1 Bayesian Optimization

Bayesian optimization is a hyperparameter optimization algorithm that is often used to optimize unknown functions that are expensive to evaluate. It tries to minimize the number of function queries by adaptively selecting the most promising hyperparameter sets for evaluation. The performance of a deep learning model can be described as an unknown function $f(\boldsymbol{x})$ of its hyperparameters $\boldsymbol{x} \in \mathcal{X}$. Thereby, each hyperparameter set is sampled from the configuration space $\mathcal{X}$ which can contain continuous as well as discrete dimensions. The key idea of Bayesian optimization is to model a relationship between hyperparameters and the objective function $f(\boldsymbol{x})$ to be optimized [23]. This is done by constructing a probabilistic model $p(f \mid D)$ of the objective function $f(\boldsymbol{x})$ where $D \in \{(\boldsymbol{x}_0, y_0), \ldots, (\boldsymbol{x}_{i-1}, y_{i-1})\}$ are the already observed datapoints obtained by $f(\boldsymbol{x})$ until iteration i. The probabilistic model captures the uncertainty about the unknown function $f(\boldsymbol{x})$ in areas where there are no observed datapoints. During Bayesian optimization, it is updated iteratively and used to guide the hyperparameter search towards regions where the objective function is likely to be optimal. To select the next set of hyperparameters for evaluation, a so-called acquisition function $a(\boldsymbol{x})$ is employed on top of the model $p(f \mid D)$. It is responsible for balancing the exploration of new areas in the configuration space and the exploitation of areas that have shown promising in the past. Bayesian optimization involves the iteration of the following four steps [23]:

(1) Select the hyperparameters that maximize the acquisition function: $\boldsymbol{x}_{t+1} = \arg\max_{\boldsymbol{x} \in \mathcal{X}} a(\boldsymbol{x})$
(2) Evaluate the objective function: $y_{t+1} = f(\boldsymbol{x}_{t+1})$
(3) Augment the datapoints: $D \leftarrow D \cup (\boldsymbol{x}_{t+1}, y_{t+1})$
(4) Refit the probabilistic model $p(f \mid D)$ with new datapoints

In general, different probabilistic models can be employed for Bayesian optimization. However, the most commonly used method is the Gaussian process which models the unknown objective function as a multivariate normal distribution over an infinite number of functions [23]. It can also be seen as a regression model which fits a set of given data points. There is an infinite number of functions that can fit a set of points, but Gaussian processes can assign a probability to each function. It outputs a mean and a covariance, which together define a distribution over the function values at any input point. The mean represents the current best guess of the true function, while the covariance quantifies the uncertainty about the function at different points in the input space. The first row in Figure 2.6 demonstrates the behavior of the predicted function

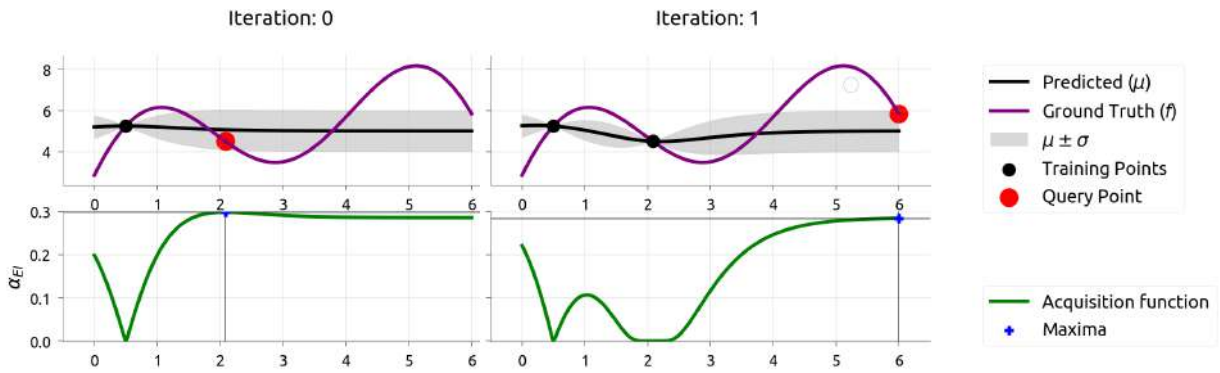


Figure 2.6: One iteration of Bayesian optimization performed on an exemplary toy function. The maximum of the acquisition function determines the query point which is evaluated next. After adding the new data point to the training set and refitting the Gaussian process model, the variance around the new point decreases. This leads to a corresponding valley in the acquisition function's shape.[5]

and the variances as more data points are added to the training set. The second row in Figure 2.6 shows the acquisition function, which in this case is the expected improvement:

$$a(\boldsymbol{x}) = \mathbb{E}\left[\max(0, \alpha - f(\boldsymbol{x}))\right] = \int \max(0, \alpha - f(\boldsymbol{x}))dp(f \mid D) \tag{2.12}$$

where α is the best value observed so far. The expected improvement depends on the probabilistic model $p(f \mid D)$. The hyperparameters which maximize its value are evaluated next. Although Bayesian optimization tries to minimize the evaluations of the unknown function, it can still be computationally expensive for deep learning models, as they can take hours and sometimes even days to converge for a single hyperparameter configuration. During the development of deep learning models, it is often desirable to find a good hyperparameter set in a limited amount of time, even if it does not correspond to the global optimum. However, if the time budget is too small, Bayesian optimization can struggle to find good configurations [23].

2.3.2 Hyperband

Hyperband is a hyperparameter optimization algorithm that was proposed in 2018 by Li et al. [51] and tackles the above-mentioned problem by taking the available resources into account. It identifies more promising hyperparameter sets early on and allocates more resources to them while discarding unpromising ones. Given a fixed budget, this adaptive resource allocation allows evaluating more configurations compared to methods that run all configurations until completion (e.g. simple random search or Bayesian optimization). In this way, Hyperband can provide orders of magnitude of speedup in finding a good parameter configuration.

[5] Source: Adapted from [4]

i	s=4		s=3		s=2		s=1		s=0	
	n_i	r_i	n_i	r_i	n_i	r_i	n_i	r_i	n_i	r_i
0	81	1	27	3	9	9	6	27	5	81
1	27	3	9	9	3	27	2	81		
2	9	9	3	27	1	81				
3	3	27	1	81						
4	1	81								

Table 2.1: Hyperband algorithm visualized for $R = 81$ and $\eta = 3$ from [51]. Successive Halving is performed $s_{max} = 5$ times for $n \in \{4, 3, 2, 1, 0\}$. The initial configuration samples in every bracket are drawn randomly and $1/\eta$ of the best-performing configurations are kept after each iteration i. The number of configurations in each iteration is indicated by n_i and the amount of resource assigned to each configuration by r_i. In each bracket, Successive Halving terminates when there is either only one configuration left ($n_i = 1$) or if the maximum budget that can be allocated for a single configuration is reached ($r_i = R$).

Hyperband is an extension of the Successive Halving algorithm which was introduced by Karnin et al. [43] in 2015. Successive Halving follows an iterative procedure that starts by randomly sampling an amount of n hyperparameter configurations. Given a total budget of B (e.g. certain amount of hours), it uniformly assigns B/n resources to each configuration. After running n different hyperparameter configurations, the performance of each model is evaluated and the worst half is discarded. The remaining $n/2$ configurations enter the next round, where again, a budget of B is uniformly allocated to each hyperparameter set. This process is repeated until one configuration is left which then constitutes the final hyperparameters.

Although the approach of successively halving the configurations shows a good performance on many problems, it has a well-known issue. The number of configurations n is a user-defined parameter and it is not clear whether it is better to choose larger or smaller values for n. Since the budget B is allocated equally, smaller values would lead to many configurations with smaller average training times and larger values to fewer configurations with larger training times. In general, if the model converges quickly or there are many bad-performing hyperparameter configurations, it is better to choose a larger n. Vice versa, it is more reasonable to choose a smaller n if the model converges slowly or if many hyperparameter configurations perform equally well. However, without that kind of prior knowledge, it is difficult to select an appropriate number of initial configurations. This is also referred to as the "n vs. B/n" problem.

Hyperband addresses this problem by performing a grid search over multiple values of n. It takes the input parameter R which defines the maximum amount of resource that should be allocated to a single hyperparameter configuration, and a parameter η which controls how many

configurations are kept after each round of Successive Halving. Both parameters together determine how many brackets are executed in total. A bracket refers to a complete run of Successive Halving for a specific n. More specifically, Hyperband executes $s_{\max} = \left\lfloor \log_\eta(R) \right\rfloor + 1$ brackets, i.e. it tries out s_{max} different values for n. In this way, Hyperband is able to find the best trade-off between the number of configurations and the allocated training time. Table 2.1 visualizes the iterations of different brackets and their resource allocation for $R = 81$ and $\eta = 3$.

Hyperband shows a strong performance for a variety of optimization problems. It typically finds good hyperparameter configurations by multiple orders of magnitudes faster than random search or Bayesian optimization. However, with larger budgets, Hyperband struggles to find the global optimum and is outperformed by Bayesian optimization (see Figure 2.7). This is because Hyperband draws the hyperparameters randomly. Hence there is a lack of guidance that could exploit good regions in the configuration space like in Bayesian optimization.

2.3.3 BOHB Algorithm

A hyperparameter optimization algorithm that combines the strengths of Hyperband and Bayesian optimization is BOHB. Introduced by Falkner et al. [21] in 2018, its goal is to achieve a strong overall performance for all amounts of resource budgets. BOHB replaces the random selection of configurations in Hyperband with a guided and more informed sampling approach using a probabilistic model. Thus, it integrates Bayesian optimization into the framework of Hyperband. Within each iteration i of Successive Halving, BOHB tracks the performances of the corresponding hyperparameter configurations and once enough data samples are collected, it starts fitting a probabilistic model (see Section 2.3.1). From there on, new hyperparameter sets are drawn based on the acquisition function. Instead of Gaussian processes which were described in Section 2.3.1, Falkner et al. used tree-structured Parzen estimators [9] to build the probabilistic model. Their training time scales only linearly with the number of datapoints compared to the cubic time in Gaussian processes.

Figure 2.7 compares the performance of BOHB with other common hyperparameter optimization methods for a neural network. It visualizes the immediate regret $r(t) = f(\boldsymbol{x}_t) - f(\boldsymbol{x}^*)$ as a function of time which computes the difference between the current solution $f(\boldsymbol{x}_t)$ at timestep t and the optimal solution $f(\boldsymbol{x}^*)$. The immediate regret for the best-found configuration is reported at every timestep. On a small to medium-time budget, BOHB is on par with Hyperband but converges faster to an optimal solution when more time is available. Compared to Bayesian optimization, BOHB finds equally good configurations in less time and even outperforms Bayesian optimization for large budgets.

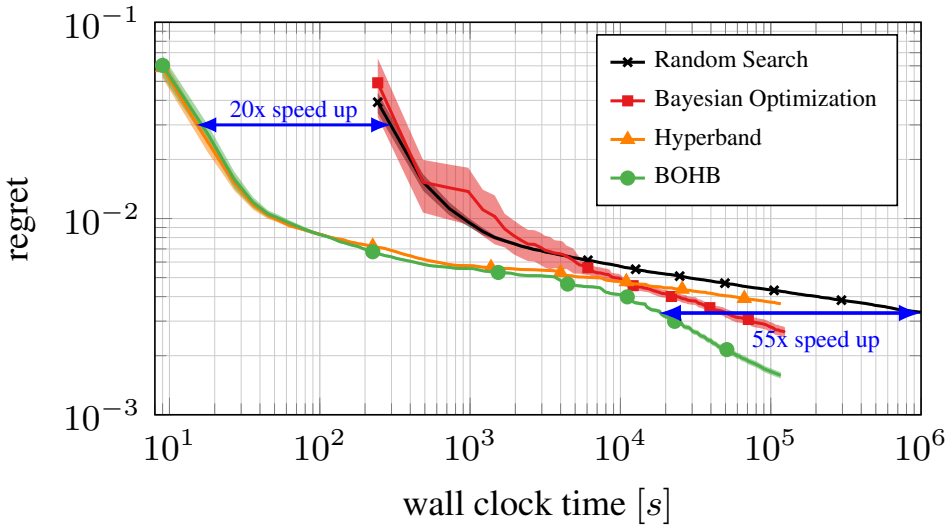


Figure 2.7: Comparison of different hyperparameter optimization algorithms on a neural network with six hyperparameters. The immediate regret is visualized as a function of time. While Hyperband yields better configurations than Bayesian optimization for small and medium time budgets, Bayesian optimization converges faster to the global optimum for larger time budgets. BOHB combines the best of both worlds: It performs equally well to Hyperband in small to medium time ranges and outperforms Bayesian optimization for larger time budgets.[6]

2.4 Weakly-Supervised Semantic Segmentation

The goal of semantic segmentation is to partition an image into regions with different semantic meanings by assigning a label to each pixel. The result is usually a dense pixel-wise segmentation map. Most semantic segmentation models use an encoder-decoder architecture [7, 14]: First, the image is passed through a series of layers, which gradually reduce the spatial resolution of the feature maps while increasing the number of channels. This process creates a more compact image representation that contains the most meaningful features. The encoded image is then passed through a decoder which again increases the spatial resolution of the feature maps while reducing the number of channel dimensions. The decoding process typically involves upsampling layers such as transposed convolutions or bilinear interpolation. A popular way to improve the model's performance is to add skip connections between the encoder and decoder of the network. For one thing, they address the problem of vanishing gradients. For another, they augment the upsampling path of the decoder with the higher resolved feature maps from the encoder. This allows the network to preserve high-resolution details and can help to refine the segmentation boundaries. An early prominent example of the encoder-decoder architecture is U-Net [69]. It influenced many subsequent works, as it was one of the first architectures using an encoder-decoder structure and also introduced skip connections into the segmentation process.

[6] Source: Figure 1 from [21]

The most common way to perform a semantic segmentation is to train a network on pixel-level ground truths. This requires a dense segmentation mask for every image and is referred to as supervised semantic segmentation. During training, the network usually minimizes a loss function that compares the predicted segmentation output with the ground truth mask for each pixel in the image. A common loss function for this task is the pixel-wise cross-entropy loss. In general, supervised methods produce more accurate segmentation maps compared to WSSS techniques, because the latter provides less spatial information about the image content. However, they are often impractical since their training requires pixel-wise annotations for every image which can be time-consuming to obtain.

Therefore, WSSS methods are often used to handle coarse training labels such as inexact, incomplete, or inaccurate annotations. Inexact labels cover larger areas than the actual ground truth, incomplete labels refer to some regions not being annotated at all, and inaccurate labels imply that some regions might be annotated with the wrong class. Common training annotations include bounding boxes, scribbles, points, and image labels [13]. While the former three provide at least some hints about the location of entities, image labels contain no spatial information at all. Due to the characteristics of the dataset in this thesis, this chapter focuses on WSSS methods that infer pixel labels from image-level annotations. WSSS algorithms can be broadly categorized into one of four categories: Expectation maximization, object proposal class inference, SSL, and MIL [13]. In the domain of natural images, SSL algorithms are predominantly used for WSSS and seem to perform better compared to the other approaches. At this time, the top five methods for WSSS on the Pascal VOC 2012 *val* dataset use SSL or extend existing SSL algorithms. However, for histopathological image data, SSL does not seem to be the best choice. It tends to perform worse compared to other WSSS methods developed for that domain [13]. For the WSSS of histopathological images, MIL algorithms are more prevalent. In fact, methodologies that are designed on natural images tend to perform poorly on histopathological image data without adaptation [13]. This aligns with the assumption that both domains may have different requirements. In the following, the fundamental concepts of SSL and MIL methods for WSSS are presented.

2.4.1 Self-Supervised Learning

The approach of SSL typically involves two steps: First, a model is trained on a pretext task with the goal to obtain informative data representations, sometimes referred to as seeds or cues. Afterwards, these representations are utilized for the actual downstream task. For WSSS, the pretext task involves training a model on image labels to produce coarse segmentation maps on pixel level. In practice, this is accomplished by training a supervised classification model and extracting the corresponding attention maps or class activation maps (CAMs) [83] for each

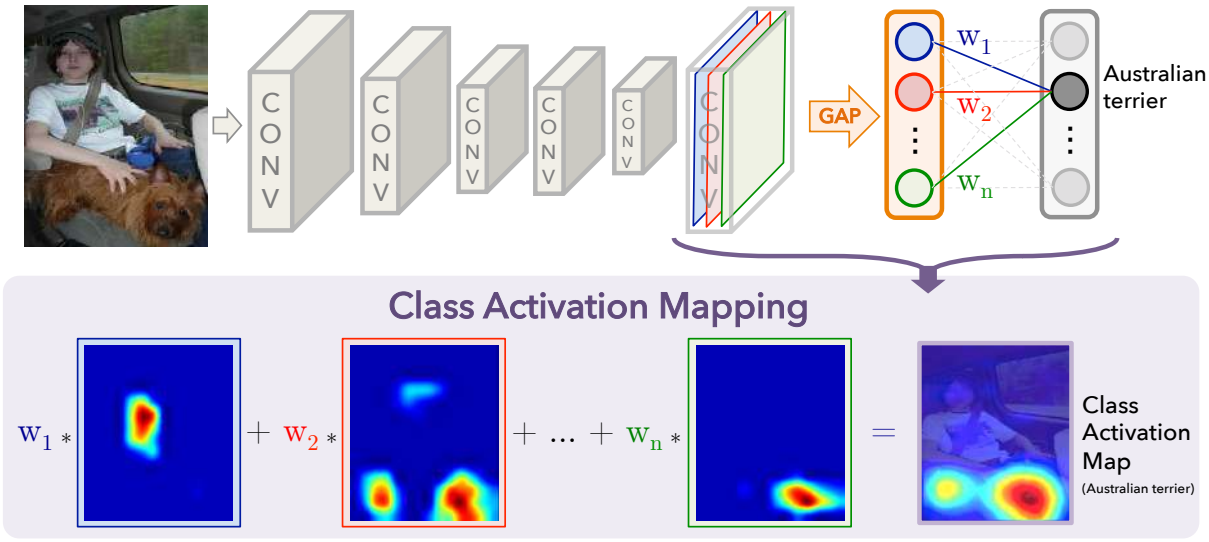


Figure 2.8: Generation of class activation maps based on a CNN trained for image classification. A global average pooling is performed on the output of the last convolutional layer forming a vector that has a corresponding neuron for each of the previous feature maps (indicated by colors). This vector is fed into a fully connected layer to produce the final classification output. The CAM for a certain class can be generated by upsampling all feature maps and taking their weighted sum with the corresponding coefficients from the fully connected layer.[7]

image. These can be used as pseudo-ground truths to train a fully-supervised semantic segmentation network, which outputs more detailed segmentation maps.

In Transformer-based classification architectures, the attention weights of the [*class*] token (see Section 2.2.4) can be used to generate cues for the downstream task. Such attention maps visualize the regions the network is attending to and contain the patterns the network has learned to recognize. In a similar fashion, CAMs are used for CNNs to highlight the regions of an input image that are most relevant for the classification decision. The CAM for a corresponding class can be calculated by upsampling the feature maps produced by the last convolutional layer to the original image size and taking their weighted sum (see Figure 2.8). The weights are determined by the learned coefficients of the following fully connected layer which produces the final classification output. Each activation map emphasizes different areas or features of the input image. The weighted sum combines the feature maps containing the content relevant to the respective class.

Although CAMs can localize the important regions to a particular class, they have a major limitation when they are used as seeds for the segmentation downstream task. In most cases, the activations do not cover entire objects but only highlight the most discriminative regions to a certain class. In addition, CAMs often fail to capture the fine-grained details of certain shapes or object boundaries. Therefore, using standard CAMs as pseudo-ground truths in SSL

[7] Source: Figure 2 from [83]

often leads to incomplete and imprecise segmentations [13]. Multiple SSL methods for WSSS focus on improving the accuracy of CAMs to create better seeds for the downstream task. One technique is to apply conditional random fields as a postprocessing step [55]. It models the relationship between adjacent pixels and enforces certain smoothness constraints to refine the CAM. Pixels that are close to each other, and similar in color and texture receive a higher probability of belonging to the same class. This can result in a more coherent segmentation, i.e. less disconnected regions, and improved object boundaries that are less jagged. Other methods that aim to improve the coverage of objects involve propagating high activations within the CAM to neighboring pixels with a similar visual appearance. This can for example be achieved through region growing [33] or random walk algorithms [1].

These methods proved to be effective in improving the quality of CAMs on natural images, leading to a better segmentation performance with SSL algorithms. However, while SSL approaches are prevalent for WSSS of natural images, they show worse performance in the histopathological image domain [13]. One explanation for this might be, that algorithms based on propagating high activations within CAMs are not suitable for histopathological images, since they often have many disconnected regions, and the morphological appearance between the fore- and background can be more subtle than in natural images. In this case, region-growing algorithms could lead to oversegmentation if they merge different clusters of cancer cells which are close to each other.

2.4.2 Multiple-Instance Learning

MIL is a popular approach in the histopathological image domain to perform WSSS. It is a supervised method that operates on bags of instances. Each bag is provided with an annotation, but the class labels for the individual instances are unknown. In practice, an image is interpreted as a bag while the instances are either patches or pixels within that image. The goal of MIL is then to derive a concept that can infer the instance labels from the bag label. This fits the WSSS scenario where every image may have one or multiple labels which indicate the presence of a class but not its exact location within the image. In the case of a binary class problem, a bag receives a positive label if it contains at least one positive instance. On the other hand, a bag is labeled negative if it does not contain any positive instance. By training a classification on a collection of bags, MIL models can induce which instances contribute to one class result or another. In this way, the model can assign labels to the individual instances. Since MIL only defines a general framework, the specific model architecture and its functionality can differ. The following related work chapter discusses different MIL methods for WSIs and Section 3.2.3 introduces the SwinMIL model which was used in this thesis.

3 Related Work

3.1 WSSS in Histopathology

This chapter discusses related work that specifically focuses on histopathological images. The previous work can be divided into two categories: The first includes methods that generate attention maps as a byproduct of a WSI classification. Although these methods are not primarily concerned with the segmentation of WSIs, the resulting attention maps can serve as seeds in the context of an SSL approach. The second category contains directly related works that focus explicitly on WSSS. The methods presented in this chapter differ in the granularity of their image labels, i.e. some methods train with labels on WSI-level while others with labels on patch-level, and in the level of detail of their segmentation results.

3.1.1 Representation Learning via WSI Classification

A recent work by Lu et al. [56] presents the weakly-supervised classification network CLAM for WSI diagnosis. It operates on WSI labels and does not require any class-specific information on pixel level. Following the MIL framework, CLAM divides each WSI into a set of patches. It processes each patch individually and aggregates all patch representations into a single WSI representation using an attention-based pooling function. The aim of CLAM is not only to perform a WSI classification but to provide more insight into which morphological features determine a particular class decision. To this end, the attention weights are used to create an attention map that highlights discriminative patches in the WSI for a particular class. This attention heatmap can also be considered as a segmentation. However, its level of detail is quite low because the network uses a patch size of 256×256 and computes only one attention score per patch. As the authors use WSIs at $20\times$ magnification, this corresponds to a resolution of approximately 128 microns per patch.

A later work that outperforms the classification performance of CLAM and provides more detailed attention maps is the one by Chen et al. [15] in 2022. They propose the self-supervised Hierarchical Image Pyramid Transformer (HIPT) model for learning slide-level representations of gigapixel WSIs. In contrast to traditional Transformer architectures which work with fixed-sized image tokens, HIPT employs token embeddings at different scales. It implements a hierarchical structure of multiple Vision Transformers: Starting with small-sized tokens, HIPT recursively performs a bottom-up aggregation of the token embeddings via global pooling to create the final slide-level representation. In this way, it captures the dependencies and interactions that may exist between structures at different resolutions of the WSI. The authors show that HIPT outperforms previous weakly-supervised approaches such as CLAM in WSI classification tasks. This was particularly evident for survival prediction where contextual in-

formation plays an important role. The attention maps at different resolutions show, that the Vision Transformers were able to capture features at different scales, such as individual cells or larger cell groups. On the lowest hierarchical level, HIPT computes the attention between 4×4 patches, which corresponds to a resolution of 2 microns per patch in the paper. However, using these high-detailed attention maps as segmentation results is not straightforward, because it is not possible to assign a unique class to each attention map. Since HIPT is trained in a self-supervised manner and does not use any WSI labels, the semantic meaning of each attention map can only be determined visually by looking at it directly. Nevertheless, this paper highlights the potential of training self-supervised Vision Transformers for segmentation tasks. This served as an inspiration to pretrain the Swin Transformer in this thesis using the same SSL approach (see chapter 3.3).

3.1.2 WSSS Methods for WSIs

Apart from the models mentioned above, which generate attention maps as a byproduct of training a classification model for WSIs, there are several methods that explicitly perform a WSSS. A popular segmentation approach is to partition a WSI into patches and classify them individually [31, 32, 81]. The recent work of Lerousseau et al. [50] introduces a model for binary tumor segmentation which classifies patches of a WSI using an advanced MIL approach. The model employs a specific prediction scheme for individual patches during training to maintain a low false positive rate. It minimizes an empirical risk function, which ensures that only a certain percentage α (e.g. 30%) of patches is classified as positive and another percentage β (e.g. 20%) as negative. This includes only the patches where the model is most confident about the class. The remaining patches are not assigned any class and are not taken into account when computing the loss function, i.e. the model is only trained with patches it is most certain about. The authors show that this method reduces the false positive rate while reliably detecting cancerous patches.

Anklin et al. [3] took a different approach by designing the graph neural network SegGini for WSI segmentation. Instead of training on WSI labels, it refines inexact and incomplete regional annotations and treats each WSI as a set of superpixels. Superpixels help to avoid jagged boundaries during segmentation since they can adapt to the structures within the WSI more accurately than patches. As part of a preprocessing step, SegGini first creates a tissue graph from the WSI, where each node represents a superpixel. A graph neural network is then applied to learn contextualized features in an iterative two-step process: First, the features of a node's neighbors are aggregated and combined with the node's own features. Afterwards, this feature vector is passed through multiple MLPs to produce the updated representation for the current node. One iteration of SegGini includes updating all nodes in the graph employing this two-step process. In this way, the network incorporates local and global relationships between superpixels into

the representation of each node. The node features are eventually classified by two different heads to obtain a segmentation of the WSI on superpixel level.

The methods mentioned so far perform segmentation by classifying patches or superpixels which are several hundred pixels in size. This results in relatively low-resolved segmentation masks. However, there are also methods that perform WSSS on a pixel level. These models typically train on labeled WSI patches and try to infer the pixel labels from the patch label. An early paper in 2017 by Jia et al. [41] introduces the model DWS-MIL which trains a CNN with deep supervision using an MIL-based approach. Deep supervision involves adding multiple loss functions at intermediate layers of the network, in addition to the final output layer. These additional supervision signals allow for better gradient flow and therefore more stable training and have shown particularly effective for semantic segmentation. In the case of DWS-MIL, the feature maps of each intermediate layer are aggregated and interpreted as separate segmentation masks for the input patch. Subsequently, a fusion layer computes the weighted average of these maps to produce the final segmentation mask. The SwinMIL model, which is applied for WSSS in this thesis, is based on a follow-up work of DWS-MIL and uses a similar architecture which is described in chapter 3.2.3. Another model for pixel-wise segmentation is CAMEL by Xu et al. [80] introduced in 2019. It addresses the problem of lacking supervision in WSSS by using an SSL approach to WSI patches. First, CAMEL generates coarse segmentation labels: It divides each WSI patch into smaller instances and infers the instance labels from the patch label using a multi-step MIL approach. In a second step, the coarse segmentation masks are then leveraged to train a fully-supervised segmentation model on WSI patches. DWS-MIL and CAMEL initially use WSI patches with high resolutions ($\approx 0.5\,\mu m$/pixel), but in both cases, the patches are downscaled before they are processed by the model which leads to relatively coarse segmentation masks.

In contrast, the authors Gu et al. [27] train a WSSS on high-resolution patches of approximately $0.5\,\mu m$/pixel without downscaling. They introduce the Siamese network HistoSegResT to generate CAMs for the semantic segmentation of WSI patches. Its backbone architecture combines the strength of CNNs to process local features with the ability of Transformers to encode long-distance relationships between image contents. This backbone is integrated into a Siamese structure with two paths. The first path processes the entire patch and generates a corresponding CAM. The second path uses the same backbone but splits the patch into multiple tiles before generating a smaller CAM for each tile. The resulting CAMs are then merged to obtain the original image size. Finally, both CAMs are fed into a reconstruction loss which is responsible for reducing the difference between the original and merged CAM. By minimizing this loss, the network is encouraged to generate consistent CAMs for the original and tiled patches of the input image. Although HistoSegResT operates on high-resolution patches with approximately $0.5\,\mu m$/pixel, it does not provide segmentation masks with a high level of detail.

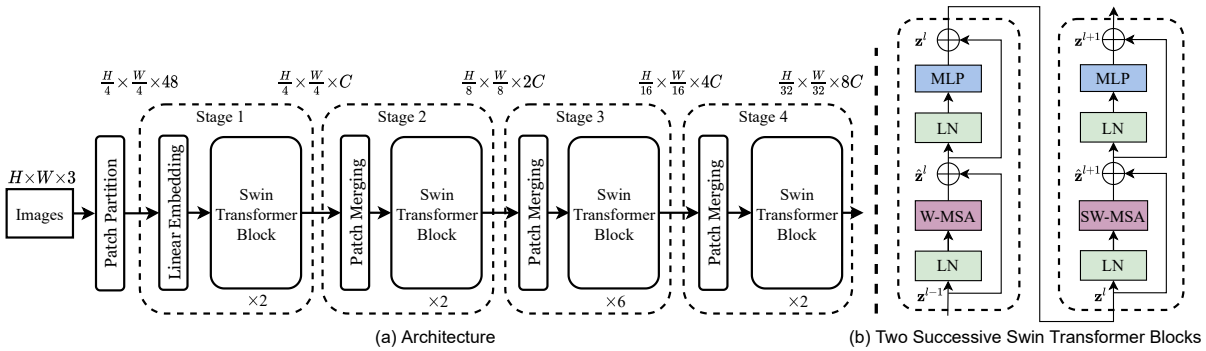


Figure 3.1: Overview of the Swin Transformer architecture. The tiny version of the Swin Transformer is depicted in (a) and comprises 4 stages. The image is first divided into 4×4 patches which are then linearly projected into a hidden subspace of dimension C in the first stage. Every following stage performs a patch merging and adopts at least one pair of Swin Transformer blocks in (b). The first block always employs a *window* multi-head self-attention (W-MSA) while the second block applies a *shifted window* multi-head self-attention (SW-MSA).[8]

The resulting masks often include background regions with no apparent tissue, i.e. HistoSeg-ResT suffers from oversegmentation.

3.2 Neural Network Architectures for Image Processing

This section describes the different neural network models that were used for the WSSS experiments in this thesis. The residual neural network was employed for the baseline model, and SwinMIL with its Swin Transformer backbone is the main model for the WSSS in this thesis. In addition, the functionality of the SSL pretraining method DINO is discussed which was used as one of the techniques to improve the performance of SwinMIL in this thesis.

3.2.1 Residual Neural Network

The ResNet architecture for image classification was introduced by He et al. in 2015 [29]. Its main idea is to facilitate the training of very deep neural networks, which were previously difficult to optimize due to the vanishing gradient problem. ResNet addresses this problem by introducing residual blocks with skip connections which ensure that the gradient signal is not diminished as it passes through the layers during backpropagation (see Section 2.2.3). These residual blocks follow the design in Figure 2.3. In this thesis specifically, the ResNet-34 architecture was employed. It has 34 layers in total and connects a series of 16 residual blocks employing 3×3 convolutions. After the 3rd, 7th, and 13th residual blocks, a max pooling layer is applied to reduce the spatial dimension of the feature maps by a factor of 2. At the end of the network, a global average pooling layer is applied to each feature map to create a

[8] Source: Figure 3 from [54]

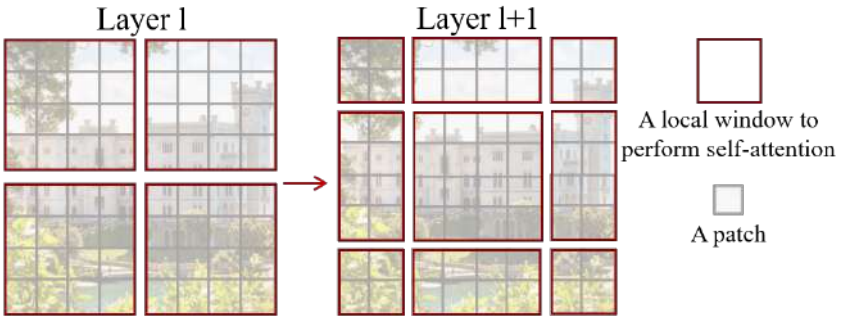


Figure 3.2: Illustration of the window partitioning scheme on a sample image. The self-attention is calculated only between patches within each local window $M \times M$ (here: $M = 4$). Attention layer l (left) employs a regular window partitioning scheme. The attention layer $l + 1$ (right) in the following transformer block shifts each window by $\frac{M}{2}$ in the horizontal and vertical directions. This allows information flow across the boundaries of the previous windows.[9]

one-dimensional feature vector. This feature vector is fed into a fully-connected layer, which produces the final classification result. An overview of ResNet-34 can be found in Figure A.1.

3.2.2 Swin Transformer

The Swin Transformer architecture was proposed by Liu et al. in 2021 [54] and was designed to overcome the difficulty in scaling Transformer-based architectures to larger image resolutions. One major limitation of previous Transformer-based networks was their quadratic computational complexity with respect to the image size. This can be illustrated by the Vision Transformer architecture: It divides the input image into a fixed number of non-overlapping patches before feeding them into the model. Because the size of each patch is fixed, the number of patches is determined by the image size. The Vision Transformer involves computing the self-attention globally between every pair of patches in the image, which requires $O(n^2)$ time complexity, where n is the number of patches. Since the number of patches is proportional to the image size, the overall computational complexity is also quadratic. As a consequence, the Vision Transfomer and similar Transformer-based architectures cannot be efficiently scaled to larger image sizes.

Swin Transformer overcomes this problem by introducing a hierarchical network design and computing the self-attention only between pairs of patches that are close to each other. Figure 3.1 shows its overall architecture. Similar to the Vision Transformer, it involves stacking a sequence of Transformer blocks. However, the self-attention layers within these blocks are implemented differently. Instead of calculating the self-attention globally, Swin Transformer blocks define local windows of a fixed size $M \times M$ and restrict the computation of self-attention to these windows. This means each token attends only to the $M \times M - 1$ other tokens within

[9] Source: Figure 2 from [54]

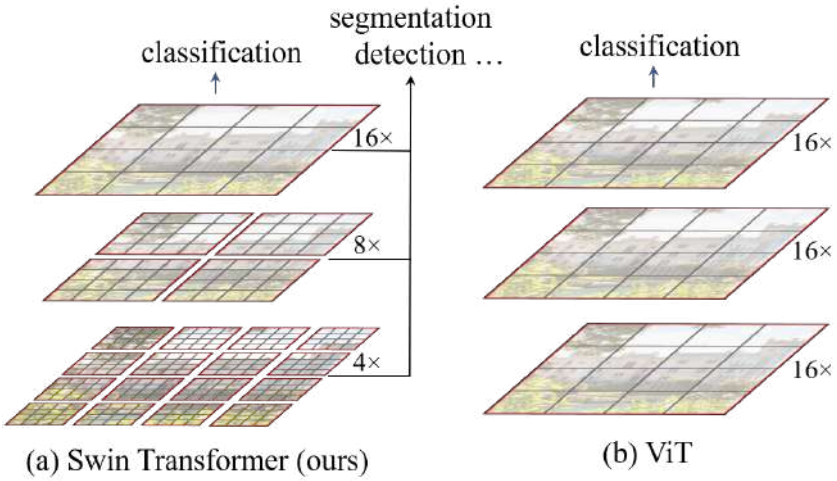


Figure 3.3: Comparison between the Swin Transformer and the Vision Transformer architecture. The Swin Transformer in (a) calculates the self-attention only within the local windows (shown in red) and has a hierarchical structure. After each stage groups of 2×2 tokens are merged and the number of tokens decreases by a factor of 4. The local window size $M \times M$ remains the same in every stage (here $M = 4$). In contrast, the number of tokens in Vision Transformer shown in (b) remains the same after each stage, and the self-attention is always computed globally between all patch tokens in the image.[10]

the same window. Since the number of tokens per window is fixed, this operation only results in linear complexity with respect to the image size.

However, a *window* attention layer by itself only allows the computation of local image features. Therefore, the Swin Transformer introduces two additional layer types which enable capturing the global context of the image. The first one is the *shifted window* attention layer. Every Swin Transformer block with a *window* attention layer is followed by a block with a *shifted window* attention layer, which shifts the previous local windows spatially (see Figure 3.2). This alternation enables information exchange across the window boundaries and combines more distant tokens with each other as the network gets deeper. The second layer type of the Swin Transformer which allows capturing long-range dependencies is the patch merging layer. After each stage, this layer combines every group of neighboring 2×2 tokens, reducing the number of tokens by a factor of 4. For example, the first patch merging layer concatenates the 4 token vectors of size C within each group into a vector of size 4C and linearly projects this vector using a fully-connected layer. The dimension of each group's output vector is set to 2C. These output vectors represent the new higher-level tokens which are then processed by the following Swin Transformer blocks. The patch merging layers are responsible for the hierarchical structure of the Swin Transformer because they reduce the spatial dimension and increase the channel dimension as the network gets deeper. Conceptually, this is similar to convolutional layers in

[10] Source: Figure 1 from [54]

CNNs since the weights of the linear projection in the patch merging layer are also shared between the 2×2 token groups. Figure 3.3 compares the hierarchical structure of the Swin Transformer with the isotropic network structure of the Vision Transformer, where neither the number of tokens nor the token dimensions change.

The Swin Transformer can efficiently handle larger image sizes and achieved state-of-the-art results in several computer vision tasks, including semantic segmentation. It serves as the backbone of the SwinMIL model which is used for the WSSS in this thesis. The next section describes the architecture of SwinMIL.

3.2.3 SwinMIL

SwinMIL is a WSSS model for histopathological images introduced by Qian et al. in 2022 [66]. The authors used the model for the binary segmentation of cancerous tissue in WSI patches. SwinMIL leverages MIL and the concept of deep supervision to learn a pixel-wise semantic segmentation from binary image-level labels. Following the framework of MIL, SwinMIL interprets each input image as a bag of instances where the bag label (image label) is given and the instance labels (pixel labels) are unknown. For training, a WSI patch was labeled cancerous if the patch contained at least one pixel belonging to a tumor cell. Otherwise, the patch was labeled healthy. Qian et al. extracted 3000×3000 patches from colorectal cancer WSIs with $40\times$ magnification (i.e. ≈ 0.25 mpp). However, to train SwinMIL, the patches were downscaled to a size of 256×256, resulting in a final patch resolution of roughly 3 mpp. In this master thesis, SwinMIL was utilized in several experiments to perform a WSSS at a higher resolution of 1.0 mpp.

Figure 3.4 gives an overview of its architecture. SwinMIL consists of a Swin Transformer encoder with 3 stages which is used to extract feature maps from the input images. More specifically, it employs the tiny version of the Swin Transformer, which is also shown in Figure 3.1a, but without the 4th stage. Furthermore, SwinMIL contains a decoding structure that generates the pixel-wise segmentation mask based on the encoded representations. For the decoding process, the feature maps are aggregated after each stage of the Swin Transformer, and the resulting side outputs are interpreted as separate segmentation masks. The exact process is the same for each of the three stages (green box in Figure 3.4): First, a 1×1 convolution is applied to the three-dimensional feature tensor to produce a two-dimensional mask. The result is then bilinearly upsampled to the size of the original image and activated using a sigmoid function to produce a probabilistic map. This map is the side output of each stage and represents a pixel-wise segmentation mask. The final segmentation mask is then generated by a fusion layer, which computes the weighted sum of all three side outputs. The weights were determined experimentally and fixed to 0.3, 0.4, and 0.3, respectively. The process of generating the final

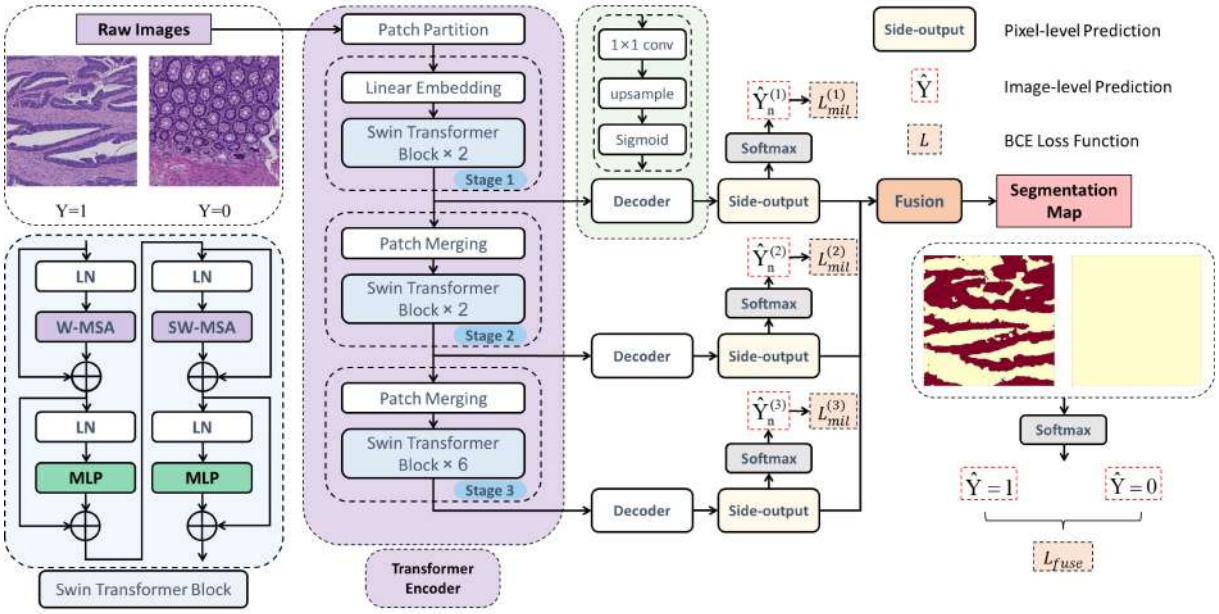


Figure 3.4: Overview if the SwinMIL architecture. A Swin Transformer backbone with three stages is used to encode the images. The final segmentation mask is created from the side outputs of all three stages to include information from different scales. Training is performed using deep supervision. In addition to the final loss function L_{fuse} after the fusion layer, a binary cross-entropy loss L_{mil} is optimized on each side output.[11]

segmentation mask from multiple stages stems from the idea of using both higher-resolution feature maps from a lower stage to increase the level of detail of the segmentation mask, as well as higher-level feature maps from a later stage to benefit from a larger context.

In order to train the network, a deep supervision approach is used. As already mentioned in Section 3.1.2, deep supervision involves adding multiple loss functions at intermediate layers of the network, in addition to the final output layer. It is used to provide stronger supervision during training, which is particularly valuable for WSSS problems since they only employ weak annotations. In the case of SwinMIL, a binary cross-entropy loss is computed after each stage based on the side output. For this purpose, each two-dimensional side output $\hat{X}^{(t)}$ at stage t is first aggregated to a scalar prediction $\hat{Y}^{(t)}$ using the generalized mean:

$$\hat{Y}^{(t)} = \left(\frac{1}{\left|\hat{X}^{(t)}\right|} \sum_{i,j} \left[\hat{X}^{(t)}(i,j)\right]^r \right)^{\frac{1}{r}} \tag{3.1}$$

where $\left|X^{(t)}\right|$ is size of the side output and $\hat{X}^{(t)}(i,j)$ is the probability value at position (i,j) of the side output at stage t. The parameter r is called sharpness and determines how close the generalized mean approaches the maximum function: $\hat{Y}^{(t)} \to \max_{i,j} \hat{X}^{(t)}(i,j)$ as $r \to \infty$. The

[11] Source: Figure 1 from [66]

authors manually set r to 4. The predictions $\hat{Y}^{(t)}$ are used to compute a binary cross-entropy loss $L^{(t)}$ for each stage t over the images in the current batch:

$$L_{mil}^{(t)} = -\sum_{i} \left(\mathbf{I}\left(Y_i = 1\right) \log \hat{Y}_i^{(t)} + \mathbf{I}\left(Y_i = 0\right) \log\left(1 - \hat{Y}_i^{(t)}\right) \right) \tag{3.2}$$

where $\mathbf{I}(\cdot)$ is the indicator function, Y_i is the ground truth label for image i and $\hat{Y}_i^{(t)}$ is the prediction for image i at stage t. In the same manner, as for each stage, a prediction is calculated for the output of the fusion layer with Equation 3.1, which is also used to compute a binary cross-entropy loss L_{fuse}. The final objective loss function is the sum of the stage losses $L_{mil}^{(t)}$ and the fusion loss L_{fuse}:

$$L = \sum_{t=1}^{3} L_{mil}^{(t)} + L_{fuse} \tag{3.3}$$

The final loss is calculated only on image labels without using any pixel-level labels. It is minimized during training using backpropagation and gradient descent.

3.3 Self-Supervised Pretraining with DINO

DINO stands for **DI**stillation of knowledge with **NO** labels and is a self-supervised learning method introduced by Caron et al. in 2021 [12]. It combines self-supervised learning and knowledge distillation to learn visual representations from images without any labels. Before describing the architecture of DINO in more detail, the concept of knowledge distillation is briefly introduced.

3.3.1 Knowledge Distillation

Knowledge distillation is a technique used to transfer information from a larger, complex model (referred to as the *teacher* model) to a smaller, simpler model (*student* model). The idea was originally introduced by Hinton et al. in 2015 [30]. It involves training the student model on the original image labels while trying to approximate the function learned by the teacher model as closely as possible. This is achieved by matching the output distributions of the teacher and student models. In mathematical terms, if $\boldsymbol{y}_{\text{teacher}}$ and $\boldsymbol{y}_{\text{student}}$ denote the output distributions of the teacher and student models respectively, the distillation process minimizes the loss function

$$\mathcal{L}_{\text{distill}} = \text{KL}\left(\boldsymbol{y}_{\text{teacher}}, \boldsymbol{y}_{\text{student}}\right) \tag{3.4}$$

where KL denotes the Kullback-Leibler divergence, a measure of the difference between two probability distributions. In general, also other terms than the Kullback-Leibler divergence can be used to measure the difference or similarity between two probability distributions (e.g. mutual information). Minimizing this loss function encourages the student model to mimic the teacher model's output distribution which contains valuable information that isn't captured by

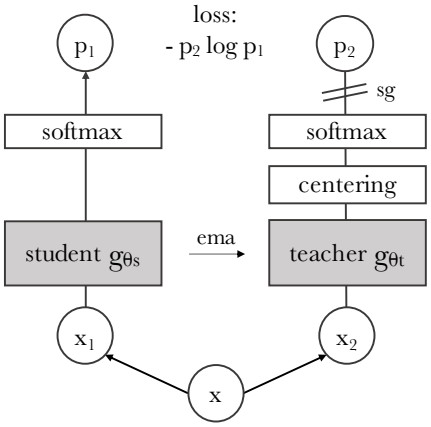


Figure 3.5: Illustration of the DINO architecture for one pair of views $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ extracted from the input image $\boldsymbol{x}$. These views are passed through the student and teacher networks, respectively. In both branches, a softmax function creates a probability distribution from the vector outputs of both networks before they are compared with each other using a cross-entropy loss. A stop gradient (sg) is applied on the teacher branch such that the loss is only optimized with respect to the student weights. The teacher weights are updated with an exponential moving average (ema) of the student weights instead.[12]

the image labels alone. Typically, the teacher model is fixed, i.e. it is pretrained in advance and its parameters do not change during the process of knowledge distillation.

3.3.2 Framework Architecture

DINO implements the basic mechanism of knowledge distillation and extends it to the case where no ground truth labels are available at all. Rather than distilling knowledge from the teacher to the student network on the exact same image, DINO encourages the student to match the teacher's output on differently augmented crops (or views) of the input image $\boldsymbol{x}$. Figure 3.5 illustrates the DINO framework for a single pair of views $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ which are passed through the student and teacher network, respectively. Both networks have the same architecture and produce one output vector each. Before measuring the similarity between these outputs, they are normalized using a softmax function. This produces probability distribution vectors over K dimensions denoted as $\boldsymbol{p}_s$ and $\boldsymbol{p}_t$ for the student and teacher network, respectively. For the student network $g_{\boldsymbol{\theta}_s}$ with parameters $\boldsymbol{\theta}_s$ and input image $\boldsymbol{x}$ the output vector $\boldsymbol{p}_s$ is calculated via

$$\boldsymbol{p}_s(\boldsymbol{x})^{(i)} = \frac{\exp\left(g_{\boldsymbol{\theta}_s}(\boldsymbol{x})^{(i)}/\tau_s\right)}{\sum_{k=1}^{K} \exp\left(g_{\boldsymbol{\theta}_s}(\boldsymbol{x})^{(k)}/\tau_s\right)} \tag{3.5}$$

where $\tau_s > 0$ is a temperature parameter that controls the sharpness of the output distribution. A similar formula applies to the probability distribution $\boldsymbol{p}_t$ which uses the teacher network $g_{\boldsymbol{\theta}_t}$

[12] Source: Figure 2 from [12]

and the temperature parameter τ_t instead. During training, both distributions are matched by minimizing a cross-entropy loss with respect to the parameters of the student network $\boldsymbol{\theta}_s$:

$$\min_{\boldsymbol{\theta}_s} H\left(\boldsymbol{p}_t(\boldsymbol{x}_1), \boldsymbol{p}_s(\boldsymbol{x}_2)\right) \tag{3.6}$$

where $H(\boldsymbol{p}_t, \boldsymbol{p}_s) = -\boldsymbol{p}_t \log \boldsymbol{p}_s$ is used as similarity measure. The parameters of the teacher network $\boldsymbol{\theta}_t$ are not changed during the minimization in Equation 3.6. However, unlike in typical knowledge distillation, they are not strictly fixed either. DINO updates the teacher parameters after each iteration with an exponential moving average of the student model parameters:

$$\boldsymbol{\theta}_t = \lambda \boldsymbol{\theta}_t + (1 - \lambda)\boldsymbol{\theta}_s \tag{3.7}$$

where λ follows a cosine schedule from 0.996 to 1 during training. This concept is related to codistillation [2], where two networks are trained together in a collaborative manner and influence each other during training. However, in DINO the teacher is not distilling from the student but updated with Equation 3.7. While training with DINO, a total of V views are generated from the input image in each iteration. The original implementation by Caron et al. generates two global views and six local views (i.e. $V = 8$). These views are randomly cropped from the input image and augmented in different ways using color jittering, Gaussian blur, and solarization. The two global views $\boldsymbol{x}_1^g$ and $\boldsymbol{x}_2^g$ cover a larger area of the image (e.g. $> 70\%$) while the local views cover a smaller area (e.g. $< 40\%$). All crops are passed through the student network. However, the teacher network processes only the two global views. The final objective function is therefore an extension of Equation 3.6 and calculated over V different views:

$$\min_{\boldsymbol{\theta}_s} \sum_{\boldsymbol{x} \in \left\{\boldsymbol{x}_1^g, \boldsymbol{x}_2^g\right\}} \sum_{\substack{\boldsymbol{x}' \in V \\ \boldsymbol{x}' \neq \boldsymbol{x}}} H\left(\boldsymbol{p}_t(\boldsymbol{x}), \boldsymbol{p}_s\left(\boldsymbol{x}'\right)\right) \tag{3.8}$$

Another important aspect of the DINO framework is the avoidance of the collapse problem. The collapse problem refers to a situation where all the input data collapses into a single or a few points in the learned representation space. In the context of DINO, the teacher and student networks could learn to produce the same output for all input images, which is a trivial and uninformative solution. To prevent this, DINO applies a centering operation on the teacher's output (see Figure 3.5) which adds a bias term c to each output vector. The bias term is updated by an exponential moving average over the previous batch outputs:

$$c \leftarrow mc + (1 - m)\frac{1}{B}\sum_{i=1}^{B} g_{\boldsymbol{\theta}_t}\left(\boldsymbol{x}_i\right) \tag{3.9}$$

where $m > 0$ is a rate parameter and B is the batch size. By adding this mean vector to each output, DINO encourages the model to learn more diverse representations and prevents all representations from converging to a single point. However, the authors showed that centering alone favors a collapse to a uniform distribution, which is why they introduced an additional sharpening in the softmax function (see Equation 3.5). It has the opposite effect of the centering operation and controls the sharpness of the output distribution. Both operations balance their effects and eventually prevent the collapse problem.

Figure 3.6: Comparison of the self-attention maps from a Vision Transformer trained with DINO versus trained in a supervised manner. The attention maps were thresholded to keep 60% of the mass of the attention map, that is the highest attention scores that collectively account for 60% of the total sum of all attention scores in the map are kept. The map from the best attention head according to the intersection over union with the image ground truth mask was chosen for this visualization.[13]

3.3.3 Performance and Benefits

The authors showed that the DINO framework enables learning powerful visual representations of images without requiring any labels. Although DINO can be used with any architecture, the Vision Transformer in particular seems to benefit from this training method. By training with DINO, a Vision Transformer was able to capture the scene layout better compared to supervised training. Figure 3.6 shows a comparison of the self-attention maps. With DINO, they capture the objects within images more completely and recognize the object boundaries more precisely. In contrast, the self-attention maps of a supervised Vision Transformer focus only on the discriminative parts of the objects and not their entirety.

Since the features of a Vision Transformer trained with DINO contain specific information about the semantic segmentation of objects, a pretraining with this method could be well suited to improve the performance of the actual WSSS task in this thesis. Therefore, one experiment within this thesis was to pretrain the Swin Transformer backbone of SwinMIL with DINO before performing the actual WSSS downstream task. An additional benefit is that the unlabelled WSIs, which represent the majority of the dataset in this thesis (see Section 4.1), could be utilized for the pretraining with DINO.

[13] Source: Figure 4 from [12]

4 Materials and Methods

This chapter first provides an overview of the histopathological image data and the preprocessing methods used in each experiment. Then, the generation of the ground truth masks for the pixel-wise evaluation of the semantic segmentation is explained. Finally, the setup of each experiment, the employed neural network models, and their training procedures are described in more detail.

4.1 Datasets

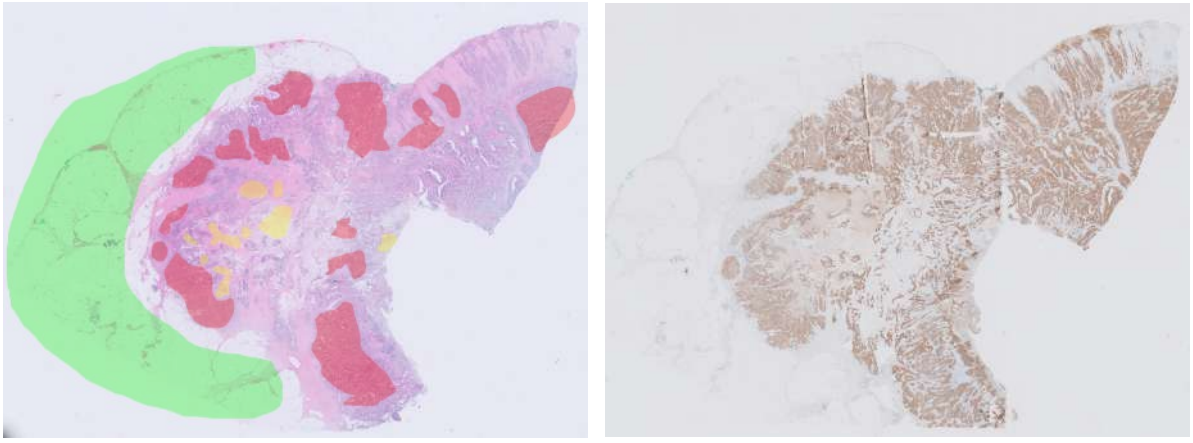
4.1.1 CancerScout-CRC

CancerScout-CRC is the primary dataset of this thesis used to generate the pixel-level ground truth masks and to train the semantic segmentation with SwinMIL. The dataset was curated by Siemens Healthineers as part of the CancerScout project [26]. It consists of 2494 WSIs of HE stained histological slides taken from 1000 patients with colorectal adenocarcinoma. In addition, the dataset contains four selected IHC stained WSIs. All tissue sections were fixed with formalin and embedded in paraffin. The patient cohort was defined by the pathology department of the University Medical Center Goettingen [25], and patients were selected from those treated between 2000 and 2020 who gave consent to be included in medical studies. Only patients with histologically confirmed adenocarcinoma of the colon or rectum were included in the dataset. Patients aged 18 years or younger and those who have received neoadjuvant therapy were excluded from the dataset. Neoadjuvant therapy refers to treatment before surgical resection of tissue.

All WSIs were initially scanned at a resolution of 0.25 µm per pixel using a PHILIPS UFS V1.8 scanner. The scanned slides were saved in iSyntax format [34] with quality preset Q2 and compressed using the PHILIPS_DP_1_0 method with a lossy image compression ratio of 15. The WSIs were then converted from iSyntax format to TIFF files using libtiff [17] and stored in a tiled pyramidal TIFF format with JPEG compression. The WSIs were downscaled to a resolution of 1.0 µm per pixel using 2×2 binning, i.e. all experiments in this thesis were performed at a resolution of 1.0 µm per pixel.

WSI-Level Annotations

Each WSI was classified as either *healthy* or *cancerous* by qualified pathologists. *Healthy* WSIs contain only healthy tissue. *Cancerous* WSIs contain cancerous tissue but may also include regions of healthy tissue and tissue otherwise afflicted by the tumor (e.g. by inflammation, scarring, or necrosis). Furthermore, the dataset contains two types of WSIs: Slides with *new* HE



(a) Annotated HE Slide (b) IHC Slide

Figure 4.1: The HE-stained WSI in (a) shows the rough regional annotations of the pathologists. The colors green, yellow, and red represent the classes *Healthy Tissue*, *No Tumor Cells*, and *Tumor Cells*, respectively. The WSI in (b) shows the same tissue section with IHC staining. The epithelial cell walls in the glandular tissue are stained brown and show potential regions of adenocarcinoma.

staining and slides with *old* HE staining. The *new* slides were cut, stained, and scanned on the same day with identical chemicals and have little color variability. The *old* slides were cut and stained up to ten years before image acquisition. They have different color characteristics compared to *new* slides and a higher color variability because the exact staining protocol, including the composition of chemicals, differed between them and aging may have affected their color. *New* and *old* slices from the same patient were resected from the same tissue specimen and are therefore located very close to each other. In total, there are four types of WSIs: *cancerous new (cnew)*, *cancerous old (cold)*, *healthy new (hnew)*, and *healthy old (hold)*.

Coarse Regional Annotations

In addition to the WSI-level labels, 300 *cancerous* WSIs were hand-labeled with coarse regional annotations by qualified pathologists. The regional annotation types are visualized in Figure 4.1a. Each annotation delineates one of the following three tissue classes:

- *Healthy Tissue* (Green)
  The region contains only healthy tissue, i.e., no live tumor cells and no tissue otherwise afflicted by the tumor.
- *Live Tumor Cells* (Red)
  The region contains tissue with live tumor cells. It does not contain any healthy tissue. However, it may also contain non-epithelial cells, which by definition cannot be affected by adenocarcinoma.

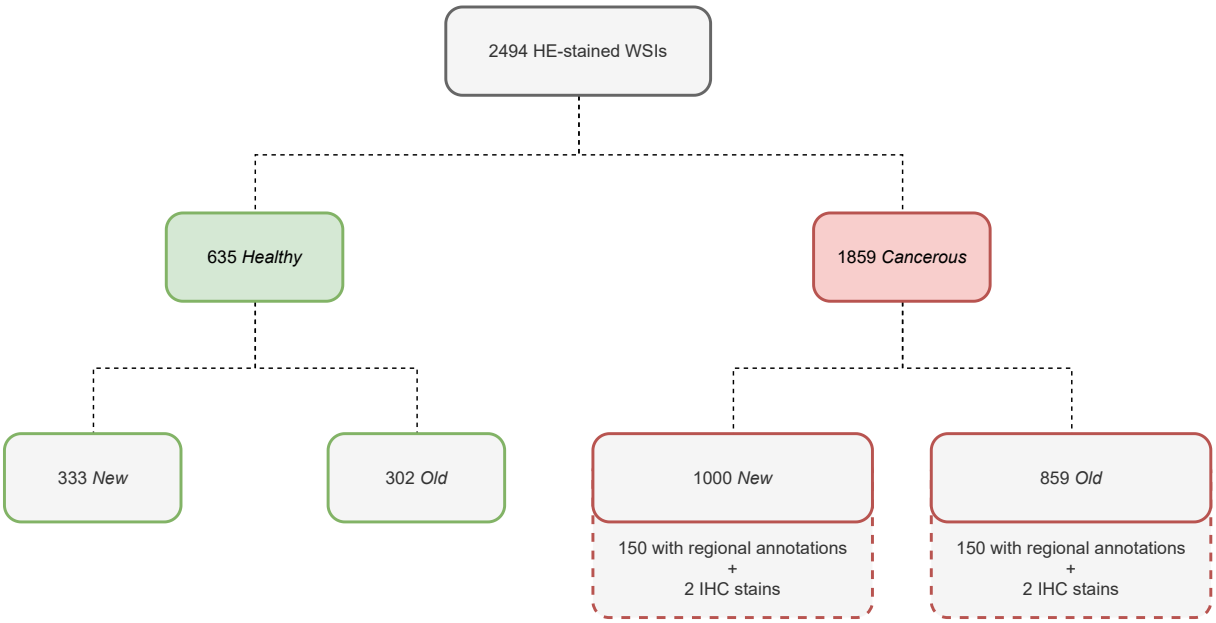


Figure 4.2: Overview of the CancerScout-CRC dataset. A WSI is either *cancerous* or *healthy* and belongs either to the category *new* or *old* HE staining. In addition, the dataset contains 150 *cancerous new* and 150 *cancerous old* WSIs with coarse regional annotations. Each of these two groups contains two HE-stained WSIs with a corresponding IHC stain.

- *No Tumor Cells* (Yellow)
  The region does not contain live tumor cells, but the tissue is not healthy. The tissue in this region is otherwise affected by the tumor, such as inflammation, scarring, or necrosis.

The regional annotations do not overlap; depending on the class, they always delineate tissue areas that meet the above-mentioned conditions. However, these annotations are not exact, i.e. they do not provide a pixel-accurate outline of the corresponding class and may include background regions without tissue. Furthermore, the annotations are incomplete, i.e. they may not fully cover a ground truth region or they may miss certain regions in the WSI. Typically, an annotated WSI still contains many unannotated tissue regions whose class is unknown. Since the goal of this thesis is to perform a binary segmentation of live tumor cells, the classes *Healthy Tissue* and *No Tumor Cells* were combined into a single class *Not Live Tumor Cells*. Therefore, the regional annotations contain either tissue of the class *Not Live Tumor Cells* (green or yellow) or *Live Tumor Cells* (red). To further simplify the terminology, patches or pixels of the class *Not Live Tumor Cells* will be referred to as *non-cancerous* and those of the class *Live Tumor Cells* will be referred to as *cancerous* in all experiments of this thesis.

To reduce the risk of mislabeling, pathologists were instructed to omit mixed or unclear areas when annotating. Furthermore, their annotations should cover the diversity of the slide. For example, instead of annotating several large regions of healthy tissue with similar morphological

appearance, it was considered more valuable to annotate a smaller region containing a minority class. Each of the 300 WSIs contains *Tumor Cells* annotations, but not necessarily annotations of the other two classes *Healthy Tissue* and *No Tumor Cells*. In addition, the size of the annotated areas can vary significantly. An overview of the CancerScout dataset can be found in Figure 4.2. The exact composition of the data used to train the models and the distribution into training, validation, and test sets is described in Section 4.4.1.

IHC-stained WSIs

To four of the 300 WSIs with regional annotations, the dataset provides an additional IHC-stained WSI. These are also available in a pyramid tiff format with a resolution of 1 mpp. In this case, the IHC staining highlights the epithelial cell walls in glandular tissue with a brown color. Since adenocarcinoma can only arise from epithelial cells, this provides information about which areas in the WSI may be affected by cancer. To account for the variability of the dataset, two IHC stains were made on *cnew* slides and the other two on *cold* slides. Figure 4.1b shows an example of an IHC-stained WSI from the CancerScout-CRC dataset.

Clinical Patient Information

Apart from labeled WSIs, this dataset also provides clinical patient information. The pathologists determined each patient's cancer grade and stage according to the TNM system of [11]. For this purpose, the tissue samples from all patients were examined histologically. In addition, the pathologists determined for each patient whether the microsatellites of their tumor cell DNA were stable. In tumor cells with *microsatellite instability (MSI)*, certain DNA segments referred to as microsatellites are unstable, meaning they are shorter or longer than in healthy tissue. Accordingly, tumor cells with an average microsatellite length are considered to be *microsatellite stable (MSS)*. In the case of colorectal adenocarcinoma, this information is an essential factor in predicting the effect of certain therapeutic methods [47]. The clinical patient information was used to split the data between the training and validation sets in the experiments (see Section 4.4.1).

4.1.2 NCT-CRC-HE-100K

The NCT-CRC-HE-100K dataset [44] was used to pretrain the Swin Transformer backbone of SwinMIL and contains non-overlapping image patches of colorectal cancer. It provides 100,000 image patches for training and 7,000 for validation. All patches have a size of 224×224 pixels at 0.5 mpp and were extracted from HE-stained tissue slides of 86 colorectal cancer patients which were provided by the NCT Biobank and the UMM Pathology Archive. The image patches were color-normalized using Macenko's method [58] and contain healthy as well as cancerous tissue. Each image was manually annotated with one of the following nine classes: Adipose

(ADI), background (BACK), debris (DEB), lymphocytes (LYM), mucus (MUC), smooth muscle (MUS), normal colonic mucosa (NORM), cancer-associated stroma (STR), and colorectal adenocarcinoma epithelium (TUM).

4.2 Preprocessing

This section addresses the preparation of the CancerScout-CRC dataset for the training of the semantic segmentation with SwinMIL. On the one hand, this includes the image registration between HE and IHC-stained WSIs, which is necessary to generate the ground truth masks for the performance evaluation (see Section 4.3). On the other hand, this section describes the image preprocessing methods that were applied to WSIs to improve their quality. They aim to increase the performance of the WSSS and allow for a more accurate model evaluation by removing artifacts from WSIs.

4.2.1 Registration of HE- and IHC-stained WSIs

Image registration is the process of aligning two or more images of the same scene. The goal is to find a spatial transformation that maps all images into a common coordinate system. Depending on the application and the nature of the images, different transformations can be considered for registration. Some examples are translation, rotation, scaling, and perspective projection.

To register a pair of HE and IHC-stained WSIs, the pixel coordinates of the IHC stain were transformed to the coordinate system of the corresponding HE stain. Since all four HE-IHC pairs in this thesis were scanned from the same viewpoint and at the same resolution of 1 mpp, the shape and size of the objects do not differ between these WSI pairs. Therefore, the pixel coordinates were transformed using only rotation and translation. This can be expressed as a rigid transformation

$$\boldsymbol{T} = \begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

where θ is the angle of rotation, and t_x and t_y are the translations in the x and y directions, respectively. The rotation and translation components of this matrix were estimated based on corresponding feature points between each WSI pair. Technically, three non-collinear point pairs are sufficient to determine the parameters of a 2D rigid transformation. However, it is advantageous to use more than three points to be more robust against noisy points and to increase the accuracy of the image alignment. Therefore, more than 20 matching points were manually selected for each HE-IHC pair. The feature points were used to set up a separate system of linear equations for every WSI pair. Finally, each system of equations was solved using the least squares method to estimate the transformation parameters.

The transformation matrix can be applied to any IHC coordinate (x, y) to transform it into the corresponding HE coordinate space. However, the two-dimensional IHC coordinates must first be transformed to homogeneous coordinates by adding a third component $w = 1$:

$$\boldsymbol{p}_{ihc} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{4.2}$$

Afterwards, the transformation matrix T can be multiplied with the homogeneous coordinate $\boldsymbol{p}_{ihc}$ to project it into the target coordinate system:

$$\boldsymbol{p}_{he} = \boldsymbol{T} \cdot \boldsymbol{p}_{ihc} \tag{4.3}$$

Finally, the projected coordinate $\boldsymbol{p}_{he}$ is converted back to Cartesian coordinates. To this end, the first two components of the vector are divided by its last component w:

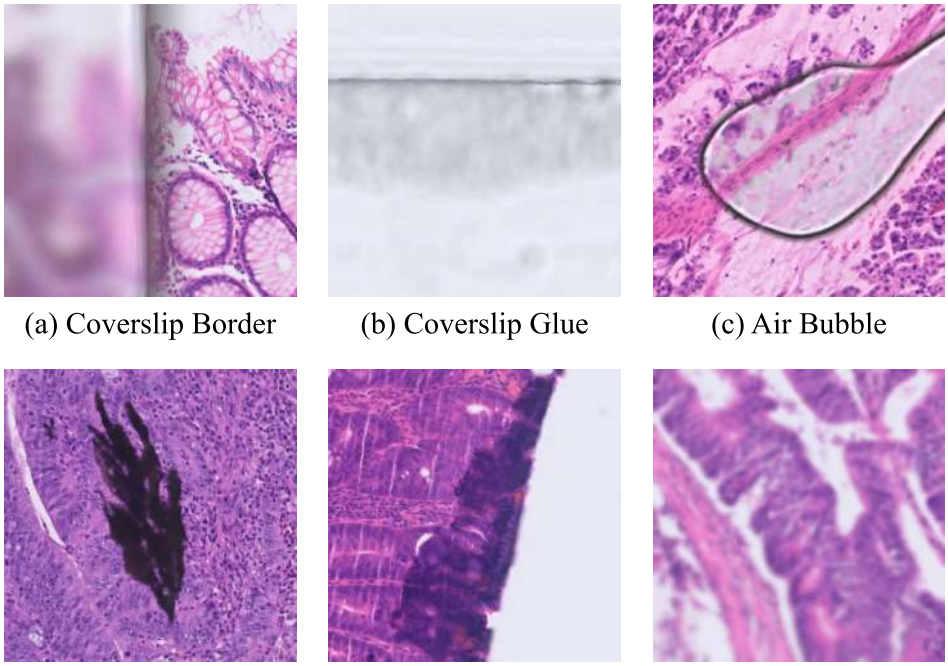
$$\boldsymbol{p}^{c}_{he} = \begin{bmatrix} \frac{x}{w} \\ \frac{y}{w} \end{bmatrix} \tag{4.4}$$

To assess the quality of the image alignment, the average projection error across all WSI pairs was determined. The projected image points differed on average by 2.64 pixels from the actual points, and the variance of the deviation was 0.52 pixels between the WSI pairs.

4.2.2 Removal of Artifacts

Artifacts in WSIs are any non-biological structures or distortions that may be introduced during sample preparation, slide digitization, or the application of digital image processing algorithms. They degrade the image quality and may negatively affect neural network performance. Figure 4.3 summarizes the main artifacts in the Cancerscout-CRC dataset. First, there are artifacts related to the coverslip on the microscope slides. Coverslips are thin, transparent pieces of glass that are placed or glued on top of the tissue sample to protect the tissue and prevent it from drying out. However, after scanning, the edges of the coverslip may become visible on the WSI (see Figure 4.3a) and the glue may leave smear marks (see Figure 4.3b). In addition, air bubbles and dirt can become trapped between the coverslip and the tissue, causing distortions in the tissue (see Figure 4.3c and Figure 4.3d). Other common artifact types in the CancerScout-CRC dataset include tissue folds or tears (see Figure 4.3e), and blurred regions (see Figure 4.3f). The former can occur during specimen preparation and leads to gaps in the tissue. The latter is related to the optics of the WSI scanner and occurs when the camera is out of focus.

An annotation software was used to manually draw rectangular bounding boxes around the above-mentioned artifacts. However, blurred regions were automatically detected in all slides (see Section 4.2.4). Regions containing artifacts were removed from the WSIs before any further processing. Depending on the WSI type and whether it was used for pretraining, training,



(a) Coverslip Border (b) Coverslip Glue (c) Air Bubble

(d) Dirt (e) Tissue Fold (f) Focus Issue

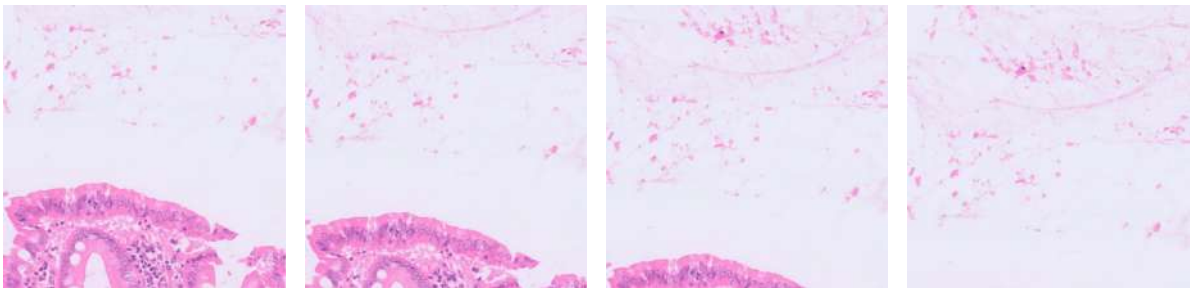
Figure 4.3: Overview of common WSI artifacts in the CancerScout-CRC dataset. The image quality can also be affected by several artifacts simultaneously. Focus issues in (f) can also be caused by the other artifacts in (a) – (e).

or evaluation of the model, different types of artifacts were excluded. The following overview describes the different cases:

(1) **Weakly-annotated WSIs for SwinMIL training**
SwinMIL was trained on the annotated tissue regions of 296 HE-stained WSIs (see Section 3.2.3). Almost none of these annotations interfere with the border of the WSIs. Therefore, coverslip artifacts at the WSI edges do not affect the training data. Other artifacts such as tissue folds, air bubbles, and dirt were not removed because they do not occur frequently and affect only small regions. Based on the amount of training data, it is assumed that these artifacts do not have a significant negative impact on the training.

(2) **WSIs for pixel-wise evaluation**
Four HE-IHC pairs were used for the final evaluation of the WSSS with SwinMIL (see Section 3.2.3). Since the evaluation was performed on pixel level and the number of images is relatively small, almost all visible artifacts were annotated manually. Approximately 15 minutes per slide were spent on annotation.

(3) **WSIs for strong negative mining**
This group contains all 635 *healthy* WSIs. They were used to mine negative exam-



(a) 70% Background (b) 80% Background (c) 90% Background (d) 98% Background

Figure 4.4: HE-stained patches with different proportions of background. The Otsu threshold was calculated on the corresponding WSI and the percentage of pixels that are considered as background is given above.

ples for SwinMIL training in one of the experiments (see Section 5.2.1). Here, the coverslip edges and the glue residues were manually removed for all of the WSIs. For time and efficiency reasons, other artifacts were only removed if they affected larger areas of the WSI and were clearly visible at the highest zoom level. About 30 seconds were spent per WSI to annotate the artifacts.

(4) **WSIs for DINO pretraining**
The DINO pretraining (see Section 3.3) employed 1700 WSIs. This includes all WSIs without coarse regional annotations. Due to the large amount of training data, it was assumed that the artifacts would not significantly affect the pretraining. Therefore, no artifacts were removed manually.

4.2.3 Background Removal with Otsu's Method

Non-tissue regions (i.e. background) were excluded from all WSIs using Otsu's method [65]. This method is a thresholding technique that aims to automatically determine the optimal threshold value for separating an image into foreground and background. To this end, the images are first converted to grayscale. Otsu's method then calculates the optimal threshold value that minimizes the intra-class variance and maximizes the inter-class variance of the pixel intensities within the image. It does this by iterating over all possible threshold values (0 – 255) and calculating the variances for each. The optimal threshold is finally used to binarize the image by classifying each pixel as either foreground or background.

Since the color spectrum may vary between WSIs, an appropriate threshold value was calculated separately for each WSI. Whether or not a region is considered as background was decided on a patch-by-patch basis because SwinMIL and the DINO pretraining approach operate on WSI patches. A WSI patch was excluded from model training if more than 80% of the pixels were

classified as background according to Otsu's method. For patches with 20% or less tissue, it was assumed that in most cases, it is not possible to decide whether the tissue is healthy, cancerous, or otherwise affected.

4.2.4 Blur Detection using Variance of Laplacian

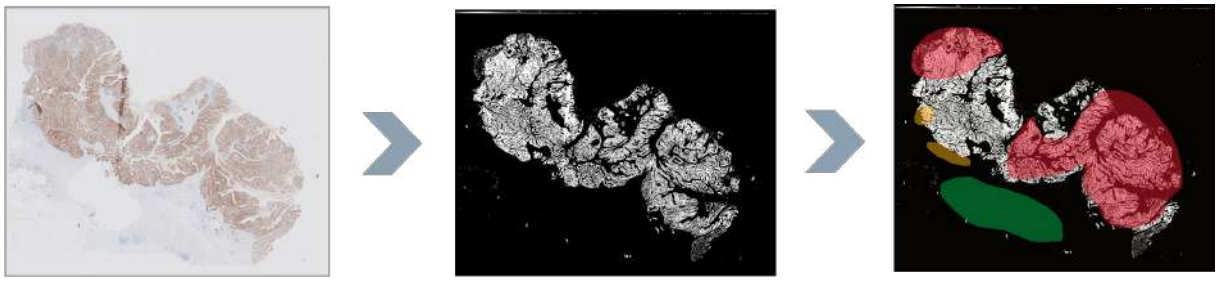
If the focal plane of the WSI scanner is not aligned with the plane of the tissue section, tissue regions may appear blurred. This can be caused by artifacts such as tissue folds or dirt on the tissue slide. If the deviation of the focal plane is too large, tissue structures or cells in the affected region may not be clearly visible. Therefore, out-of-focus regions were automatically detected and removed from all WSIs. For this purpose, a WSI was converted into a grayscale image and divided into patches. Each WSI patch was convolved with a Laplacian filter, and the variance of the pixel values was calculated afterwards. Patches with a variance less than 80 were considered blurry and excluded from the WSI. The basic idea behind this method is that the sharpness of an image can be quantified by measuring the local variation in pixel intensities. The Laplacian operator produces an output that represents the second derivative of the input image, highlighting areas of rapid intensity change, such as edges and corners. Therefore, the variance measures the amount of variation in the Laplacian image. A higher variance indicates more pronounced edges and corners in the image, i.e. a sharp image. In contrast, blurred images have low variance because the intensity values tend to be more similar between neighboring pixels.

4.3 Evaluation of WSSS Results

WSSS models are typically trained with coarse annotations and produce higher resolved segmentation masks with finer details. Therefore, a coarse ground truth mask as used for training is not sufficient to determine the quality of the segmentation result. In the case of SwinMIL, image-level labels are used for training to produce a high-resolution segmentation mask on pixel level. However, since the CancerScout-CRC dataset contains only WSIs with coarse regional annotations and no pixel-wise ground truth masks, it is not straightforward to evaluate the performance of SwinMIL in a quantitative way.

4.3.1 General Idea

The idea in this thesis is to generate a surrogate signal for the pixel-wise ground truth masks, using the four available IHC-stained WSIs and their corresponding coarse regional annotations. The IHC stains provide an accurate mask of epithelial cells and indicate regions that may be potentially affected by adenocarcinoma. Any cells that are not marked on the IHC stain cannot



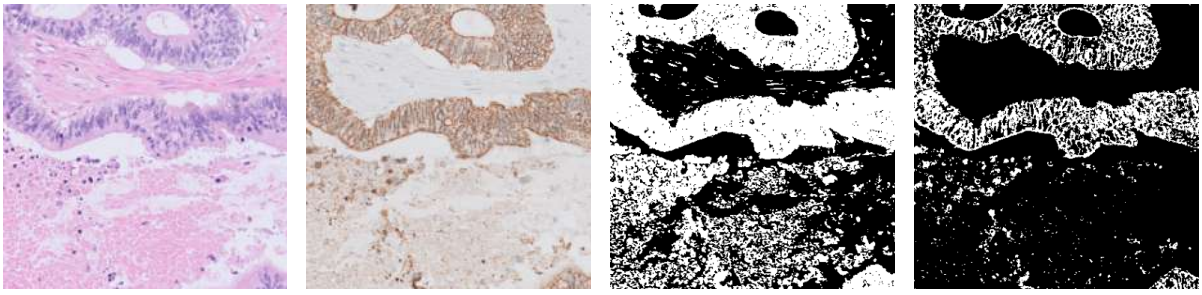
(a) IHC Slide (b) Binary IHC Slide (c) Registered Overlay

Figure 4.5: Illustration of the approach for generating ground truth masks on pixel level. First, an IHC slide is transformed into a binary mask as shown in (b) using image processing algorithms. Next, the binary mask is overlaid with the corresponding regional annotation mask as shown in (c). The regions where both masks overlap can be used for the pixel-wise evaluation of WSSS algorithms.

be affected by adenocarcinoma. On the other hand, the coarse regional annotations indicate which regions definitely contain live cancer cells (red annotations in Figure 4.1a) and which do not (green and yellow annotations in Figure 4.1a). However, the red annotations also contain subregions with non-epithelial cells, i.e. cells that cannot be affected by adenocarcinoma.

Figure 4.5 illustrates the idea behind the generation of the pixel-wise ground-truth masks. It involves generating a binary mask for each IHC-stained WSI and overlaying it with its regional annotation mask. The union of the two masks provides the regions that can be used to evaluate SwinMIL on pixel level: The intersection of red regions with the binary IHC mask represents live cancerous epithelial cells. In contrast, the intersection of green or yellow regions with the binary IHC mask represents epithelial cells that are definitely not live cancer cells. In other words, each pixel in the regions where the two masks intersect can be classified with certainty as *Live Tumor Cells* or *Not Live Tumor Cells*. All other regions where the two masks do not overlap have uncertain pixel labels and therefore cannot be used for the evaluation of SwinMIL on pixel level. The four HE-stained WSIs in the Cancerscout-CRC dataset, which have both a corresponding IHC stain and regional annotations, are therefore used in this thesis to evaluate the segmentation performance of SwinMIL within all experiments.

However, for the described procedure to work technically, accurate binary masks must be generated from the IHC-stained WSIs. This cannot be achieved by trivial thresholding for two reasons: First, although IHC staining marks the epithelial cell walls in deep brown, the cell nuclei still appear bluish-white and are therefore difficult to separate from the non-tissue background. However, the nuclei should also be included in the binary IHC mask. Second, the IHC stain also contains brownish regions that do not represent epithelial cells and are considered artifacts or noise. These regions should not be included in the final mask. Both observations are shown in Figure 4.6b. The results of two basic thresholding techniques are shown in Figure 4.6c



(a) HE Patch (b) IHC Patch (c) Binary Mask 1 (d) Binary Mask 2

Figure 4.6: WSI patch of epithelial tissue. The cell nuclei in the upper half of the IHC stain (b) are bluish-white and surrounded by brown-marked cell walls. The lower half of the patch contains almost no epithelial cells. However, the IHC staining still contains brownish artifacts for most of this region. The binary masks in (c) and (d) are the result of basic thresholding.

and Figure 4.6d. The former was obtained by converting the IHC patch to a grayscale image and performing a binary thresholding of the intensity values. This mask contains many artifacts that do not represent epithelial cells. For the latter, the IHC patch was converted to HSV color space and only pixels within a certain range of brown color were included in the mask. Although this mask contains fewer artifacts, it does not include the epithelial cell nuclei. The following section describes the procedure and the image processing algorithms used to generate accurate binary masks for the epithelial tissue in IHC slides.

4.3.2 Generation of Binary IHC Masks

The IHC stain was first converted to a grayscale image, such that brown tones were represented in different shades of gray. A morphological opening was then applied to remove high-frequency details such as noise from the image, resulting in a more homogeneous image (see Figure B.1). It also helped to reduce the bright elliptical areas associated with the cell nuclei between the epithelial cell walls. The morphological opening was performed with an 8×8 kernel using an ellipse as a structural element. In the next step, eight bilateral filters were applied to the image, one after the other. This was used to smooth the image while maintaining a clear separation between the epithelial tissue and the background. By applying the bilateral filter several times, the image gradually became smoother and converged to an almost binary image with sharp edges. Different values were tested for the parameters $\sigma_{intensity}$ and σ_{space} of the bilateral filter. Their effects on the mask quality are shown in Figure B.2 and Figure B.4, respectively. Finally, the parameters $\sigma_{intensity} = 15$ and $\sigma_{space} = 15$ were used for all eight bilateral filters. Afterwards, the image was transformed into a binary mask by applying simple thresholding: All pixel values ≤ 170 were set to white (epithelial cells), and all pixel values > 170 were set to black (background). This threshold was determined experimentally on several patches and produced nearly identical masks for all values between 150 and 170 (see Figure B.3). Thus, the

final binary mask is not too sensitive to the threshold value. After thresholding, several small artifacts that do not belong to the epithelial tissue remain in the binary mask. These were removed using a connected-component analysis (see Figure B.5). The components were modeled with a 4-way connectivity and all components < 100 pixels were removed from the image. The resulting binary mask was then used to generate the ground truth mask for pixel-wise evaluation.

4.3.3 Mask Quality and Limitations

In most cases, the generated masks correctly separate the epithelial tissue from the background and other cell types in the IHC stains. Figure 4.7 shows a selection of 400×400 patches where the presented algorithm works well from a qualitative perspective. The masks in these examples correctly include the cell walls and the nuclei of the epithelial cells. Large to medium-sized circular background regions surrounded by epithelial tissue are also correctly excluded, and the boundaries to the background are accurate. However, there are a few cases where the generated binary mask is inaccurate and contains regions without epithelial tissue. Examples of such cases are shown in Figure 4.8. The mask for the IHC patch in (a) incorrectly includes background due to a shadow. The mask in (b) contains brown artifacts that do not represent epithelial tissue. This indicates that the algorithm has difficulty excluding artifacts that appear dark on the IHC stain or have an intense brown color similar to that of epithelial tissue. The masks in (c) - (f) erroneously include bluish immune cells. Thus, the algorithm also appears to have difficulty identifying the boundaries of epithelial tissue precisely when it is surrounded by many bluish immune cells. Such errors in the binary mask can lead to inaccuracies in the final pixel-wise ground truth mask and affect the quantitative numerical results.

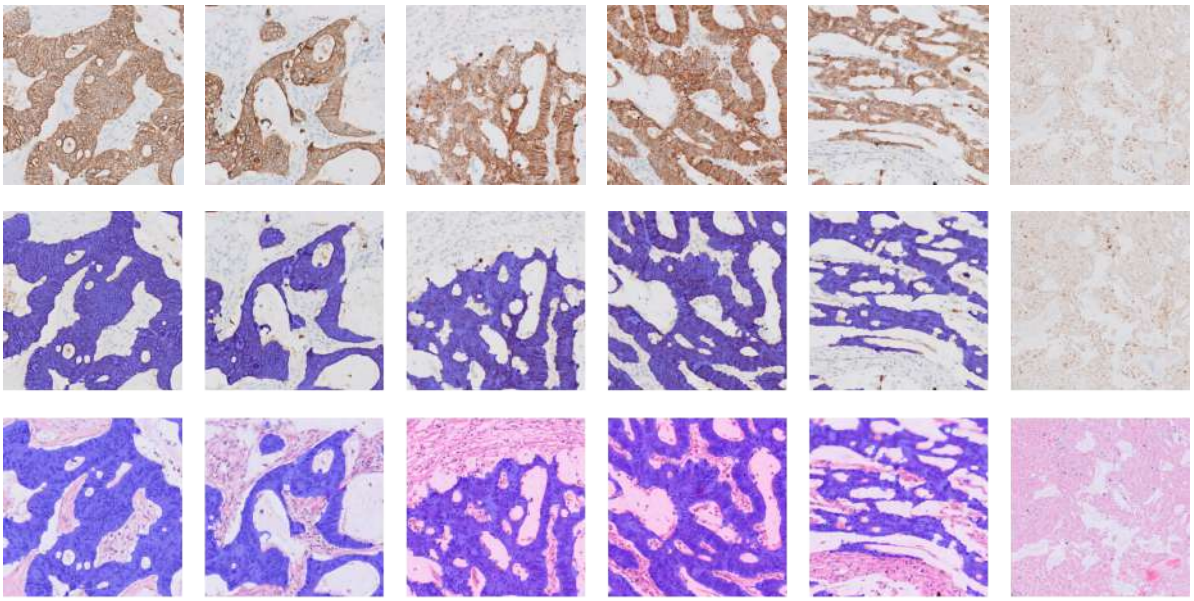


Figure 4.7: Successful examples of binary masks of epithelial tissue. The first row shows the original IHC patches. The second and third rows showcase the ground truth masks superimposed on the IHC and the corresponding HE patch.

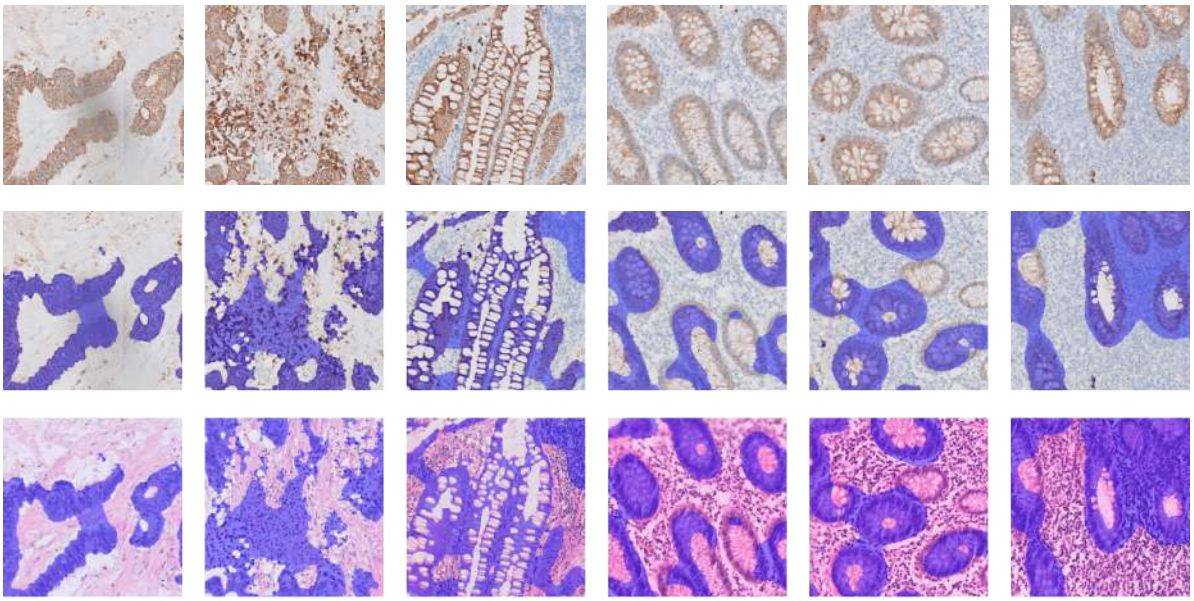


Figure 4.8: Unsuccessful examples of binary masks of epithelial tissue. The first row shows the original IHC patches. The second and third rows showcase the generated ground truth masks superimposed on the IHC and the corresponding HE patch.

4.4 Experiment Descriptions

This section describes the setup of the different experiments performed in this thesis. The purpose of the experiments was to evaluate and improve the WSSS of *cancerous* cells on high-resolution WSIs. Initially, a simple supervised ResNet-34 baseline was implemented to provide a qualitative reference for the WSSS experiments with SwinMIL. Following this, four different experiments were conducted using the SwinMIL model. One of the WSSS experiments evaluated the effect of pretraining the Swin Transformer backbone with DINO on the segmentation performance. In order to find appropriate parameters for DINO, four parameter configurations were tested specifically for the pretraining of the Swin Transformer.

4.4.1 Data Composition for WSSS Experiments

In all four WSSS experiments, the SwinMIL model was trained and evaluated on patches extracted from HE-stained WSIs of the CancerScout-CRC dataset. Since SwinMIL relies on image-level labels, only WSI patches with known labels were employed. On the one hand, this includes patches extracted from *healthy* WSIs. Such patches were labeled as *non-cancerous* since *healthy* WSIs contain only healthy tissue. On the other hand, this includes patches extracted from the coarsely annotated regions of *cancerous* WSIs: Patches from green or yellow regions were labeled *non-cancerous*, and patches from red regions were labeled *cancerous*. This fits the general MIL framework, where a patch is considered *cancerous* if it includes at least one pixel that is labeled *cancerous*. To reduce the number of incorrect labels, patches with less than 80% overlap with the green or yellow region were excluded. Such patches include too many unannotated tissue areas and could potentially contain pixels of the *cancerous* class. Therefore, they cannot be labeled with certainty as *non-cancerous* according to the MIL framework. Patches outside the coarse regional annotations were also not used for training or evaluation of SwinMIL, as their tissue class is unknown. For each WSSS experiment, a training, validation, and test set were created.

Training and Validation Set

The exact composition of the training and validation sets depends on the specific experiment. Figure 4.9 gives an overview of the dataset properties for the different experiments which are further described in Section 4.4.3. In general, the experiments employ different WSI types: Three out of four experiments use only patches from the annotated regions of *cancerous* WSIs, while one experiment (hard negative mining) also includes patches from *healthy* WSIs. Furthermore, the number of patches may vary between the experiments because they use different patch sizes. For each experiment, 80% of the extracted WSI patches were utilized for training and 20% were reserved for validation. To ensure a similar patch distribution within the training

and validation sets, a stratified random split was employed. The patches were split according to the following attributes:

- Target class of the patch: *cancerous*, *non-cancerous*
- WSI type from which the patch originated: *cnew*, *cold*, *hnew*, *hold*
- MSI status of the patient: *MSI*, *MSS*
- Primary tumor size (pT) of the patient: *1, 2, 3, N/A*

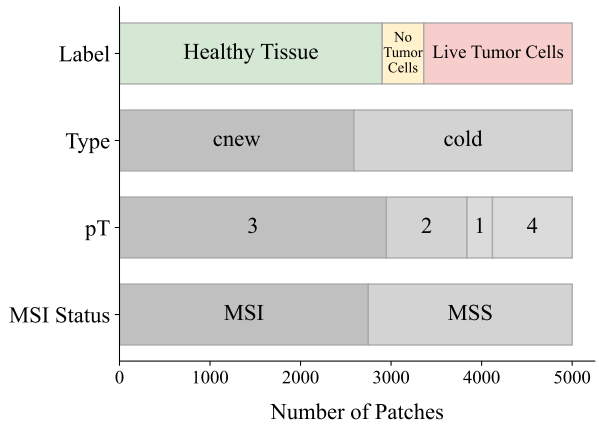
In addition, it was ensured that the patients between the training and validation sets were mutually exclusive, i.e. no patches from the same WSI were present in both sets at the same time. This serves to prevent potential leakage of patient-specific features that could be exploited by the model during prediction. SwinMIL was validated on patch level, i.e. all metrics for hyperparameter optimization were computed on patch labels.

Test Set

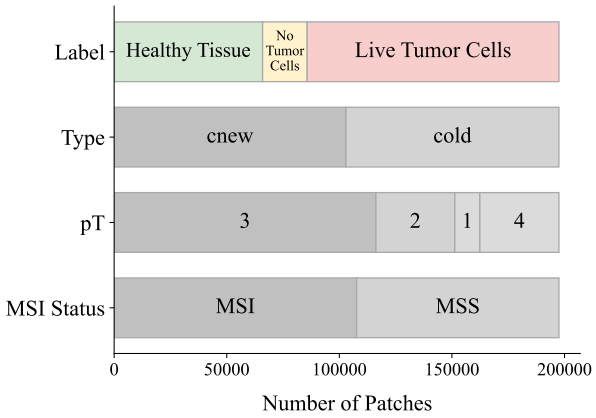
The WSIs in the test set were manually selected and were identical for the baseline model and all four WSSS experiments with SwinMIL. Unlike the validation set, it was used to evaluate the final segmentation performance of SwinMIL on pixel level. It consists of patches extracted from 10 *healthy* WSIs (5 *hnew*, 5 *hold*) and from the regional annotations of the 4 *cancerous* WSIs (2 *cnew*, 2 *cold*) for which the pixel-wise ground truth was generated. Since the number of extracted patches depends on the patch size, the exact composition of the test set is specified with the respective experiment description in Section 4.4.3. The WSIs from the test set were used to calculate the segmentation metrics on pixel level and the employed metrics are described in more detail in Section 4.5.2.

4.4.2 Baseline Model: Patch Classification with ResNet-34

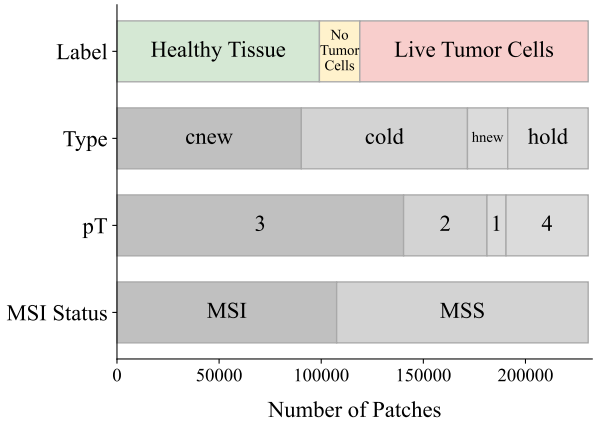
Serving as a baseline for the WSSS experiments with SwinMIL, a supervised ResNet-34 model was trained to perform WSI segmentation through patch classification. This helped to identify potential problems with the data and the modeling pipeline early on and served as an initial visual reference for the level of detail that could be achieved by classifying *cancerous* patches. The baseline model was trained and evaluated on approximately 200K patches of size 400×400 extracted from the annotated regions of 296 *cancerous* WSIs. Figure 4.9b illustrates the representation of the patch attributes which were used to split the patches between the training and validation sets. The data split was performed in the same stratified fashion as for the WSSS experiments. The test set consisted of 2658 *cancerous* and ~23K *non-cancerous* patches extracted from the 4 *cancerous* and the 10 *healthy* WSIs in the test set.



(a) Experiment 1: Low-Resolution



(b) Experiment 2: Reference Model
Experiment 3: DINO Pretraining



(a) Experiment 1: Low-Resolution

Figure 4.9: Overview of the patch attribute representation in the datasets for different experiments. The number of patches depends on the number of employed WSIs and the patch size. Each of these datasets was split into a training and a validation set within the respective experiment.

The ResNet-34 was optimized for a simple binary cross-entropy loss on patch labels using SGD with momentum and weight decay. In addition, a learning rate decay with a factor of 0.1 was applied every seven epochs. The hyperparameter optimization was performed on the learning rate, batch size, and weight decay using the BOHB algorithm and is further described in Section 4.5.3.

4.4.3 WSSS with SwinMIL

Four different experiments were conducted using SwinMIL. Unless otherwise stated, the Swin Transformer backbone was always initialized with weights from the ImageNet pretraining which were published by the authors of the Swin Transformer [53]. As a first experiment, SwinMIL was applied to low-resolution patches with large spatial contexts. The purpose was to evaluate

the effect of a large spatial context on the pixel-wise segmentation performance and to contrast the qualitative difference between high- and low-resolution segmentation masks. All subsequent experiments were performed on high-resolution patches of 1 mpp. The second experiment focused on determining the optimal patch size for training SwinMIL on high-resolution patches. On the one hand, the patch should provide enough context for the model to segment cancer cells; on the other hand, large patch sizes might make it more difficult for the model to infer the correct location of *cancerous* pixels from the patch label. Following this experiment, the patch size was fixed and two further experiments were conducted to improve the segmentation performance of SwinMIL. One involved the pretraining of the SwinMIL backbone with DINO, and the other consisted of training SwinMIL with negatively mined examples. For each WSSS experiment with SwinMIL, a hyperparameter optimization was performed using the BOHB algorithm, which is discussed in more detail in Section 4.5.3. The following overview gives a more detailed description of the four experiments:

(1) **Experiment 1: Low-Resolution Implementation**
Similar to the original work by Qian et al., this experiment was performed on 3000× 3000 patches, which were scaled down to a size of 256×256, before being used for training, validation, or testing. Since the original WSI resolution in this thesis is 1 mpp, this resulted in a patch resolution of approximately 12 mpp. Figure 4.9a illustrates the patch attributes. A total of 2902 *cancerous* patches and 2099 *non-cancerous* patches were extracted from the regional annotations of 296 *cancerous* WSIs and split between training and validation set in a stratified fashion. The pixel-wise evaluation was performed on 42 *cancerous* and 385 *non-cancerous* patches extracted from the 4 *cancerous* and the 10 *healthy* WSIs in the test set.

(2) **Experiment 2: Finding Optimal Patch Size**
To find the optimal patch size for high-resolution patches, SwinMIL was trained on eight different patch sizes between 200×200 and 900×900 pixels. Patches larger than 900×900 pixels were not evaluated due to memory constraints on the GPU. The sizes were chosen in increments of 100 pixels (i.e. 200×200, 300×300, etc.). By treating the patch size as an additional hyperparameter during optimization, each size was evaluated multiple times with different model hyperparameters. Depending on the patch size, the number of patches for training, validation, and testing differed for each run and is therefore not further specified.
After this experiment, it was found that there were no significant performance differences between different patch sizes. Therefore, the patch size was fixed to 400×400 and another hyperparameter optimization run was performed with BOHB. The performance of the best model on patch size 400×400 served as a reference for the following two experiments, which aimed to improve the segmentation performance of SwinMIL. Therefore, this model will be referred to as the reference model for

the remainder of this thesis. The patch attributes of the dataset used for training and validation of the reference model are shown in Figure 4.9b. The pixel-wise evaluation was performed on 2658 *cancerous* and ~23K *non-cancerous* patches extracted from the 4 *cancerous* WSIs and the 10 *healthy* WSIs in the test set.

(3) **Experiment 3: Improving Performance by Pretraining with DINO**
This experiment aimed to improve the performance of SwinMIL by pretraining its Swin Transformer backbone with DINO before performing the actual WSSS task. Four different configurations were evaluated for the pretraining (see Section 4.4.4) and the best one was chosen to initialize the weights of the Swin Transformer backbone of SwinMIL for this experiment. All weights of SwinMIL were then finetuned on 400×400 patches from the training set. The patch attributes of the dataset used for training and validation in this experiment are shown in Figure 4.9b. The pixel-wise evaluation was performed on 2658 *cancerous* and ~23K *non-cancerous* patches extracted from the 4 *cancerous* WSIs and the 10 *healthy* WSIs in the test set.

(4) **Experiment 4: Improving Performance by Hard Negative Mining**
Since the reference model in Experiment 2 was found to have a large number of false positives in the healthy slides, the idea in this experiment was to reduce them by hard negative mining. It involved selecting ~45K *non-cancerous* patches from 625 *healthy* slides that the SwinMIL model misclassified as *cancerous* and adding them to the training set as negative examples. This should help SwinMIL to better distinguish these negative examples from positive examples to improve its segmentation performance. The best SwinMIL model after hyperparameter optimization on 400×400 patches from Experiment 2 was used to infer ~1 million patches from 625 *healthy* WSIs on pixel level. The 10 WSIs from the test set were excluded from this procedure. Next, the top 4% of patches with the most false positives on pixel level were added to the training set. This included ~22.5K *hnew* and ~22.5K *hold* patches. The resulting distribution of the dataset in this experiment for training and validation is shown in Figure 4.9c. The pixel-wise evaluation was performed on 2658 *cancerous* and ~23K *non-cancerous* patches extracted from the 4 *cancerous* WSIs and the 10 *healthy* WSIs in the test set.

4.4.4 Swin Transformer Pretraining with DINO

For the WSSS Experiment 3 with SwinMIL, the Swin Transformer backbone was pretrained with the SSL method DINO. For this purpose, four different parameter configurations were manually tested for DINO. The weights of the best configuration were then selected to initialize the Swin Transformer for the subsequent WSSS with SwinMIL. In each of these four configuration experiments, the Swin Transformer was pretrained on approximately 4.7 million unlabeled

400×400 patches from 625 *healthy* and 1559 *cancerous* WSIs. This included all *cancerous* WSIs from the CancerScout-CRC dataset that do not have coarse regional annotations and were therefore not used for training, evaluation, or testing of the SwinMIL model. In addition, the 10 *healthy* WSIs from the test set were excluded from the DINO pretraining. The following overview describes the four parameter configurations. Since DINO has about 30 model parameters, only those that were changed in the respective experiments will be explained:

(1) **Original DINO Implementation**
In the first experiment, the default model parameters were adopted from the original work by Caron et al. Since the DINO implementation in the original work employed natural images and the Vision Transformer architecture, this experiment examines its applicability to the setting in this thesis: The aim was to evaluate how well the DINO pretraining works on the Swin Transformer with histopathological image data.

(2) **Decreasing Dimensionality of Network Outputs**
In this experiment, the dimensionality of the two output vectors generated by the heads of the student and teacher networks was decreased. These vectors are passed through the softmax function during pretraining and finally compared with each other using the cross-entropy loss. Roughly speaking, the smaller the two output vectors, the faster the DINO network can match them and converge to a minimum loss. Convergence time is an important factor to consider, since the original DINO implementation trains for quite a long time. However, the downside of smaller output vectors is that the network is less able to capture complex patterns in the input data. For the DINO pretraining in this thesis, two different output dimensions were considered. In one trial the vector dimension was set to 2^{14} and in a second trial to 2^{15}. Both values are smaller than the output dimension in the original implementation, which is 2^{16}.

(3) **Removing Weight Normalization in Last Layer**
In this attempt, the weight normalization of the last layer in the heads of the student and teacher networks was omitted. According to the authors of the original paper, this can lead to better model performance for smaller networks but at the same time carries the risk of making the training unstable. The authors omitted the weight normalization when pretraining the small version of the Vision Transformer in their work to increase its accuracy. Since the SwinMIL model in this thesis employs the tiny version of the Swin Transformer, removing the weight normalization during pretraining could also be advantageous.

Each configuration was trained for 24 epochs and evaluated every 5 epochs on the CancerScout-CRC and NCT-CRC-HE-100K datasets. The goal of the evaluation was to track the ability

of the Swin Transformer models to extract meaningful features from the WSI patches during training. To this end, two different classification tasks were performed based on the feature vectors generated by the Swin Transformer models.

Evaluation of Feature Quality via k-NN Classification

The first classification task involved labeled patches from the CancerScout-CRC dataset. Specifically, 200K patches of size 400×400 from the coarsely annotated regions of the 296 *cancerous* WSIs were used for the classification task (see Figure 4.9b). As in the WSSS experiments, these patches were first divided into training and validation sets. For each configuration experiment, these patches were encoded with the pretrained Swin Transformer. A k-NN classifier was then trained on the resulting feature vectors from the training set. Afterwards, the accuracy of the k-NN classifier was evaluated by classifying each patch in the validation set as *cancerous* or *non-cancerous*.

For the second classification task, another k-NN classifier was trained on 100K patches from the NCT-CRC-HE-100K dataset for each configuration. In this task, each patch was assigned to one of nine possible classes, making it more challenging than the previous binary classification on the CancerScout-CRC dataset. All patches were also encoded in advance using the pretrained Swin Transformer. The accuracy of the k-NN classifier was evaluated on the 7K patches from the validation set of the NCT-CRC-HE-100K dataset.

Eventually, the k-NN classification proved to be more informative on the NCT-CRC-HE-100K dataset than on the CancerScout-CRC dataset. Therefore, the weights of the pretrained Swin Transformer that provided the highest classification accuracy on the NCT-CRC-HE-100K dataset were selected to initialize SwinMIL in WSSS Experiment 3.

Visualization of Features

The features obtained from the best pretraining configuration were visualized to evaluate the representations of the Swin Transformer qualitatively after DINO pretraining. To this end, the dimensionality reduction techniques principal component analysis (PCA) [59] and Uniform Manifold Approximation and Projection (UMAP) [60] were used to reduce the high-dimensional feature vectors into a two-dimensional space.

PCA is a linear method that orthogonally transforms the original coordinates into a new set of coordinates known as principal components. These principal components are the directions in data space along which the original data points vary the most. The first principal component accounts for the largest variance and the second principal component (which is orthogonal to the first) accounts for the second largest variance, and so on. The idea behind using PCA for feature visualization is to project the high-dimensional feature vectors into a lower-dimensional space

using the principal components. In this thesis, PCA is applied to the feature vectors extracted from the patches of the NCT-CRC-HE-100K dataset. The first two principal components are then used to project the feature vectors into a two-dimensional space. Unlike PCA, UMAP is able to capture more complex, non-linear structures. It uses the concept of a manifold which can be thought of as a surface embedded in a high-dimensional space and tries to learn this surface in a way that is faithful to the local and global structure of the data. That means, it tries to preserve the distances and relationships between data points when projecting them from a high-dimensional space to a lower-dimensional one. As a result, UMAP can often produce more meaningful visualizations than PCA [60]. In this thesis, the parameters $n\_neighbors = 17$ and $min\_dist = 0.1$ were employed for UMAP after several rounds of experimentation. The parameter *n_neighbors* controls the balance between preserving the local and global structure of the data. A smaller value of *n_neighbors* emphasizes preserving the local structure over the global structure, and vice versa, a larger value gives more weight to the global structure, making the output less sensitive to fine-grained structures in the data. The parameter *min_dist* controls the minimum distance allowed between points in the low-dimensional representation. A smaller value allows points to cluster more tightly together. This preserves fine detail but at the risk of capturing more noise. A larger value causes the points to spread out, resulting in a more evenly distributed embedding with less detail and noise.

4.5 Neural Network Training

4.5.1 Data Augmentation

To increase the amount of data and make the model more robust to color variations in the input patches, a number of image transformations were applied during training. The image patches were randomly flipped horizontally and vertically, and rotated by a multiple of 90 degrees with a probability of 0.5. In addition, a random color jitter was randomly applied to each patch: Brightness, contrast, saturation, and hue were adjusted with a probability of 0.5 in the range of 10%, 20%, 20%, and 5%, respectively.

4.5.2 Metrics

F1 Score

The main metric used to measure the performance of the SwinMIL model in this thesis was the F1 score. It is calculated as the harmonic mean of precision and recall:

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{4.5}$$

with

$$\text{precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

and

$$\text{recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

The precision is the proportion of correct positive predictions among all positive predictions made by the model. High precision means that a model makes few false positive predictions. The recall on the other hand is the proportion of correct positive predictions among all actual positive instances in the dataset. High recall means that a model correctly identifies most positive instances, while low recall means that a model misses many positive instances. In general, there is often a trade-off between precision and recall: Improving precision may come at the cost of lower recall, and vice versa. The F1 score balances both scores.

In this thesis, two different F1 scores were considered: The *patch F1 score* and the *pixel F1 score*. The former was computed on patch labels and used to validate SwinMIL and the ResNet-34 baseline for different hyperparameter configurations. The latter was calculated on pixel labels and was exclusively used to evaluate the SwinMIL predictions on the four *cancerous* WSIs from the test set which are provided with pixel-wise ground truth masks. The *pixel F1 score* does not include any of the other 10 *healthy* WSIs from the test set, as they do not contain cancerous tissue. The *patch F1 score* on the validation set was used to guide the hyperparameter search with BOHB in all experiments. This is because using the *pixel F1 score* would leak information from the pixel-wise ground truth masks into the training process, violating the assumption of WSSS that the model is trained only with spatially less informative labels.

Specificity

The specificity is also known as the true negative rate and measures a model's ability to identify negative instances correctly. It is defined as:

$$\text{specificity} = \frac{\text{True Negatives}}{\text{True Negatives} + \text{False Positives}} \tag{4.6}$$

A high specificity indicates that the model makes few false positive predictions. On the other hand, a low specificity means that the model misclassifies many negative instances as positive. In this thesis, the specificity was computed on pixel level and is denoted as *pixel specificity*. It was used to evaluate the false positives in the 10 *healthy* WSIs from the test set.

Hausdorff Distance

The Hausdorff distance is a measure of the difference between two sets of points in a metric space. In the context of semantic segmentation, the Hausdorff distance can be used to measure the dissimilarity between two segmentation masks in terms of their shape and boundaries. The

one-sided Hausdorff distance $\tilde{H}(A, B)$ is defined as the maximum distance between any point in one segmentation mask $A = \{a_1, a_2, \ldots, a_n\}$ and its nearest point in the other segmentation mask $B = \{b_1, b_2, \ldots, b_m\}$:

$$\tilde{H}(A, B) = \sup_{a \in A} \inf_{b \in B} d(a, b) \tag{4.7}$$

where $d(a, b)$ is the distance function between points a and b. In this thesis, the Euclidean distance was used. Furthermore, the bidirectional Hausdorff distance $H(A, B)$ can be calculated by additionally considering $\tilde{H}(B, A)$ and taking the maximum of both one-sided Hausdorff distances:

$$H(A, B) = \max\left(\tilde{H}(A, B), \tilde{H}(B, A)\right) \tag{4.8}$$

The bidirectional Hausdorff distance was used in this thesis to evaluate the boundaries of the segmentation masks generated by SwinMIL. Like the *pixel F1 score*, the Hausdorff distance is computed on pixel level and only on the 4 *cancerous* WSIs from the test set, as the *healthy* slides have no *cancerous* pixels. Since one WSI consumes quite large amounts of memory and the segmentation results of SwinMIL are generated patch by patch, the Hausdorff distance was also calculated on a patch basis. That is, Equation 4.8 was applied to pairs of patches, calculating the Hausdorff distance between a segmented patch A and its corresponding ground truth patch B. To obtain a single Hausdorff distance for one WSI, the average across all patch pairs was calculated.

4.5.3 Hyperparameter Optimization

A hyperparameter optimization with BOHB was performed for the ResNet-34 baseline model and for each of the SwinMIL models in the four WSSS experiments. The hyperparameters in each experiment were optimized for the *patch F1 score* on the validation set and included the learning rate, batch size, and weight decay. In Experiment 2 with SwinMIL, the patch size was also included in the search space. The resource for BOHB was set to the number of training iterations and one iteration was defined as training a model with 100K patches. The hyperparameter optimization in each experiment involved training 100 models with different hyperparameter configurations. Each model was trained for a different number of iterations but for a maximum of $R = 6$ iterations (i.e. 600K patches). This maximum was found to be sufficient to achieve convergence for the ResNet-34 baseline as well as SwinMIL. The reduction factor η for each bracket within BOHB was set to 2, resulting in a total of 3 brackets. Tree Parzen estimators were used in BOHB to fit the probabilistic model for Bayesian optimization.

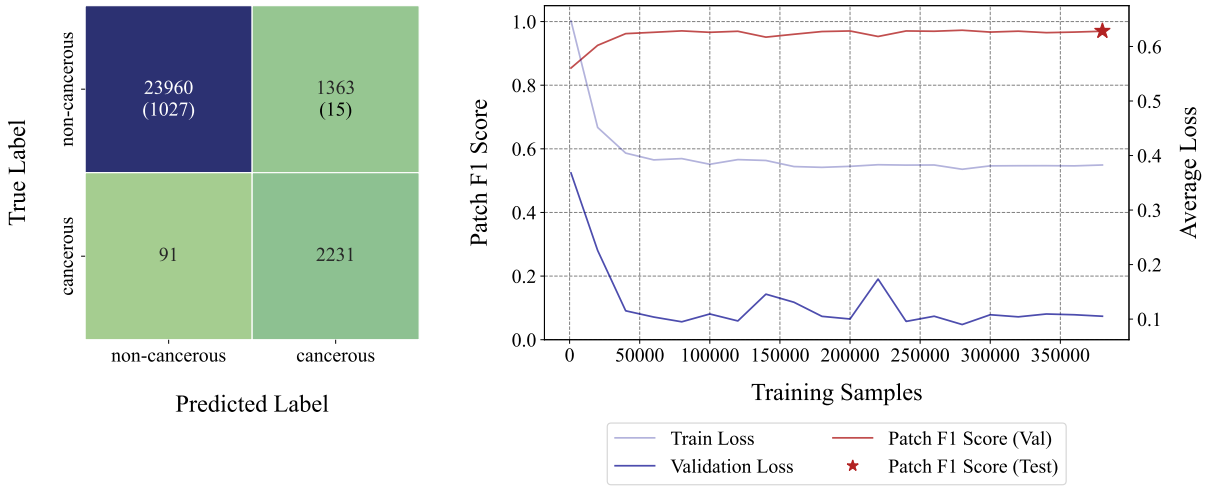
4.5.4 Software Framework

All programming work in this master thesis was done in Python 3.8. The models were implemented with PyTorch 1.13 and OpenCV 4.6 was used to preprocess the WSI patches. To effi-

ciently extract and load patches from the large giga-pixel WSIs during model training, pyvips 2.2 was used. The hyperparameter optimization was performed with RayTune 2.2. It provided an implementation of the BOHB algorithm and a live overview in the terminal to follow the optimization process. The quantitative results of the experiments were visualized using TensorBoard 2.13, which provided a clear overview of the calculated metrics and visualized the correlations between hyperparameters and model performance. Additionally, pandas 1.5 and NumPy 1.23 were used to perform exploratory data analysis and to apply common matrix or tensor transformations. The image labeling tool EXACT was used to manually annotate the artifacts in the WSIs and to select point coordinate pairs for WSI registration.

4.5.5 Hardware

The experiments in this master thesis were conducted on different hardware. The hyperparameter optimizations for the ResNet-34 baseline model and for the SwinMIL models in all four experiments were mainly performed on four NVIDIA GeForce GTX 1080Ti GPUs with 11GB memory and partly on two NVIDIA GeForce RTX 2080 GPUs with 8GB memory. The hyperparameter optimization of the SwinMIL model with BOHB took an average of seven days per experiment. The pretraining of the Swin Transformer with DINO was performed on the Siemens Healthineers Sherlock supercomputer. Each of the four pretraining experiments was run on a node with eight NVIDIA V100 GPUs with 16GB memory and took four days to complete.



(a) Confusion Matrix (b) Metrics

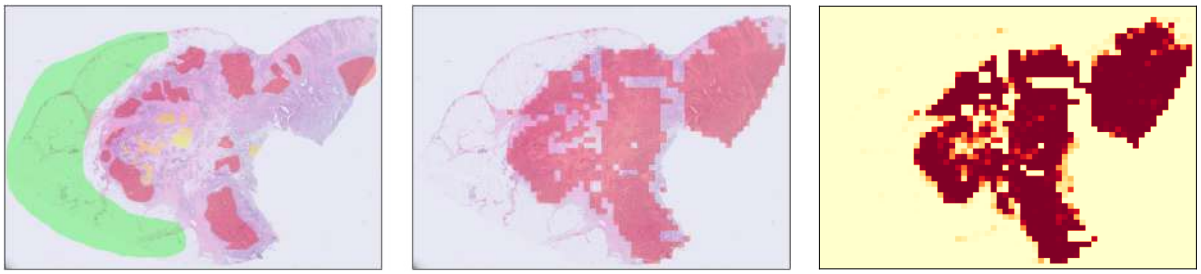
Figure 5.1: Numerical results of the best baseline model. The confusion matrix in (a) shows the performance of the model on the test set. The number in brackets denotes the amount of patches originating from *cancerous* WSIs. In (b), the evolution of the losses and the *patch F1 score* during the training of the model is shown.

5 Results and Evaluation

This chapter presents the results of the baseline model and the four described WSSS experiments, as well as the results of the four pretraining trials with DINO. The section on the baseline model and the WSSS experiments is structured into a description of the quantitative numerical results and an illustration of the qualitative prediction masks on the WSIs. The section on self-supervised pretraining with DINO presents the k-NN classification results of the four trials and a visualization of the features of the best trial.

5.1 Baseline Model: Patch Classification with ResNet-34

The baseline model from the hyperparameter optimization with the highest *patch F1 Score* on the validation set was selected for evaluation on the test set. Figure 5.1a shows the confusion matrix which provides insight into the patch classification results of the best baseline model. To add more context, the numbers in the brackets for the *non-cancerous* patches indicate how many patches originate from the 4 *cancerous* WSIs in the test set. From the confusion matrix, it appears that the model performs relatively well in classifying both cancerous and non-cancerous patches. It classified 23960 *non-cancerous* patches correctly and misclassified 1363 as cancerous. For the *cancerous* patches, the model correctly classified 2231 patches as cancerous but incorrectly identified 91 patches as non-cancerous, which indicates that *cancerous* patches are not often missed. This results in a final *patch precision* of 0.946, a *patch recall* of 0.996, and

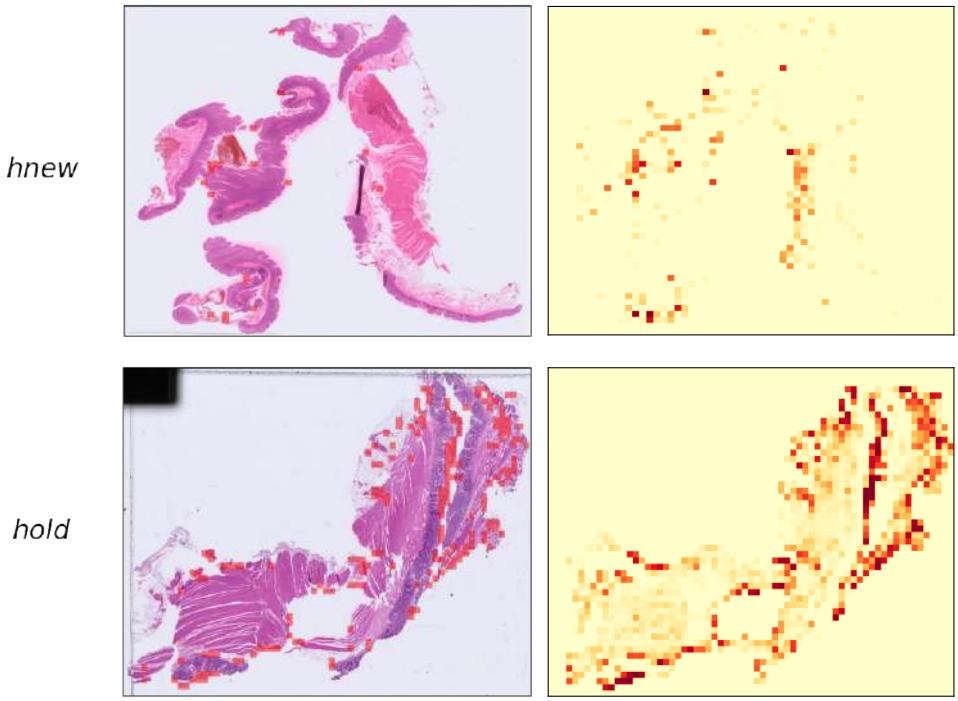


(a) Coarse Ground Truth (b) Prediction (Binary) (c) Prediction (Heatmap)

Figure 5.2: Qualitative prediction mask of the best baseline model on one *cnew* WSI from the test set. The coarse annotation masks are shown in (a). The binary prediction of the model is superimposed on the WSI in (b) and a raw heatmap of the prediction values is presented in (c).

a *patch F1 score* of 0.971. Figure 5.1b shows the trends of the loss curves and the *patch F1 score* over the course of the training. In total, the model was trained with 400K samples, and for every 20K samples, the average losses and the *patch F1 score* were calculated. However, the first evaluation was already performed after 1000 training samples. The training loss decreases from an initial value of 0.647 to 0.383 after 100K samples and stays relatively constant from there. The *patch F1 score* shows a similar behavior but increases only up to 50K samples and then remains relatively stable, hovering around a value of 0.96 – 0.97. This shows that the baseline model already converges after 50K samples to its best performance and does not improve with further training. Although the spikes in the validation loss curve after 80K samples might indicate an overfitting problem, this has no significant impact on the *patch F1 Score*.

Looking at the qualitative prediction masks of the baseline model, their low resolution becomes apparent. Figure 5.2b shows an example of the predictions on a *cnew* WSI of the test set. The classification of patches of size 400×400 leads only to a low level of detail. In addition, the figure shows that the model reliably detects all cancer regions, i.e., has few false negatives. This is consistent with the high *patch recall* of 0.996. However, the model also classifies many patches as *cancerous* outside the coarse regional annotations. Such predictions could not be evaluated because the ground truth in these regions is unknown. Figure C.1 shows a more complete picture of the baseline model predictions on all four *cnew* WSIs in the test set. Looking at the *healthy* WSIs it becomes clear that the baseline model has some problems with false positives. This is also indicated by the *patch precision* of 0.946, which is slightly lower in relation to the *patch recall*. Figure 5.3 shows an example of the model predictions on two *healthy* WSIs of the test set. It seems that mainly healthy epithelial tissue is classified as false positive. In addition, the model seems to make more false positives in the *hold* slides than in the *hnew* slides. This becomes even clearer when comparing the predictions on all *cnew* slides from the test set (see Figure C.2) with the predictions on all *cold* slides (see Figure C.3).

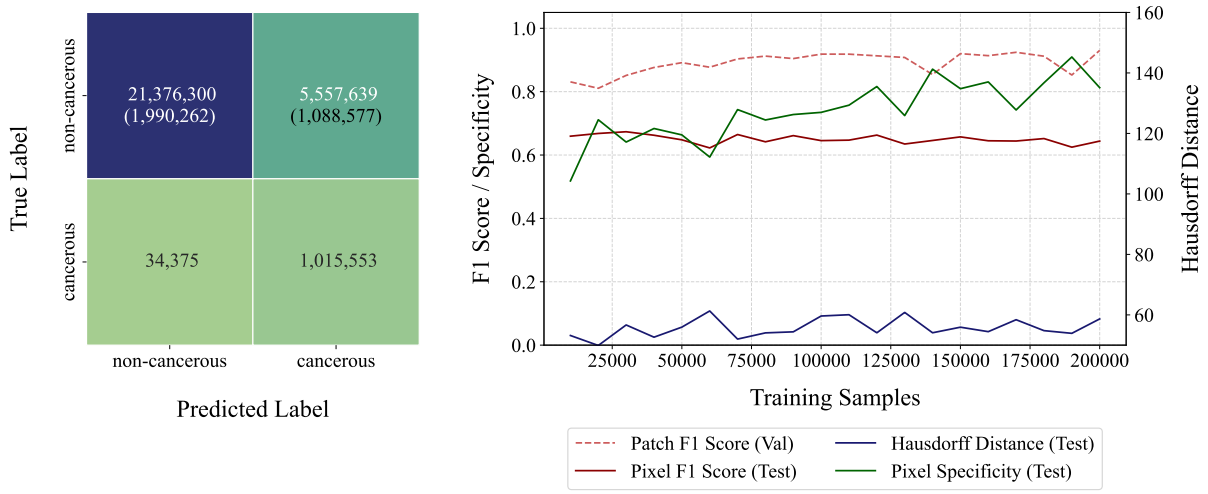


(a) Prediction (Binary) (b) Prediction (Heatmap)

Figure 5.3: Qualitative prediction mask of the best baseline model on one *hnew* and one *hold* WSI from the test set. The binary prediction of the model is superimposed on the WSI in (a) and a raw heatmap of the prediction values is presented in (b).

Experiment	*Patch F1 Score*	*Pixel F1 Score*	*Pixel Specificity*	*Hausdorff Distance*
(1) Low-Resolution	0.930	0.644	0.812	**58.6**
(2) Reference	**0.975**	**0.701**	0.946	108.9
(3) DINO	0.949	0.687	0.868	101.4
(4) Negative Mining	0.879	0.676	**0.999**	107.9

Table 5.1: Summary of the results from all experiments. For each experiment, the metrics of the model with the highest *patch F1 score* on the validation set are presented. The best result across all experiments is marked in bold for each metric.



(a) Confusion Matrix (b) Metrics

Figure 5.4: Numerical results for the best low-resolution SwinMIL model. The confusion matrix in (a) shows the performance of the best checkpoint on the test set. The number in brackets denotes the amount of patches originating from *cancerous* WSIs. In (b), the evolution of the metrics during the training of the model is shown. The *Pixel F1 score* on the test set was calculated for all checkpoints, but not used for model selection.

5.2 WSSS Experiments with SwinMIL

5.2.1 Quantitative Results

In the following, the quantitative results of the four WSSS experiments are described and compared with each other. For each experiment, a hyperparameter optimization was performed and the model with the highest *patch F1 score* on the validation set was used for the final pixel-wise evaluation on the test set. This model will be referred to as the best model in each experiment in the following. Table 5.1 summarizes the numerical results of the best models across all four experiments. It should be noted that all metrics with the prefix *pixel* were calculated at pixel level on the test set and metrics with the prefix *patch* were calculated at patch level on the validation set. Furthermore, the *pixel F1 score* was computed using only *cancerous* WSIs and the *pixel specificity* was computed using only *healthy* WSIs.

Experiment 1: Low-Resolution Implementation

Figure 5.4a shows the confusion matrix of the segmentation results for the best low-resolution SwinMIL model. The model has a high true positive rate, i.e. it detects *cancerous* pixels quite well. However, there are a considerable number of false positives, i.e. the model often incorrectly identifies *non-cancerous* pixels as *cancerous*. This is reflected in a low final *pixel F1 score* of 0.644 for the *cancerous* slides and a *pixel specificity* of 0.813 for the *healthy* slides.

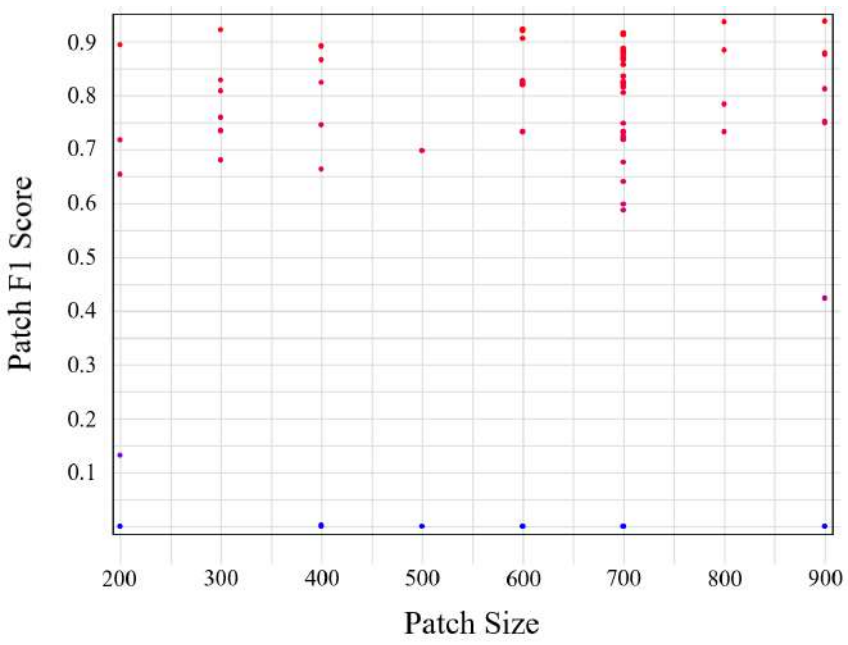
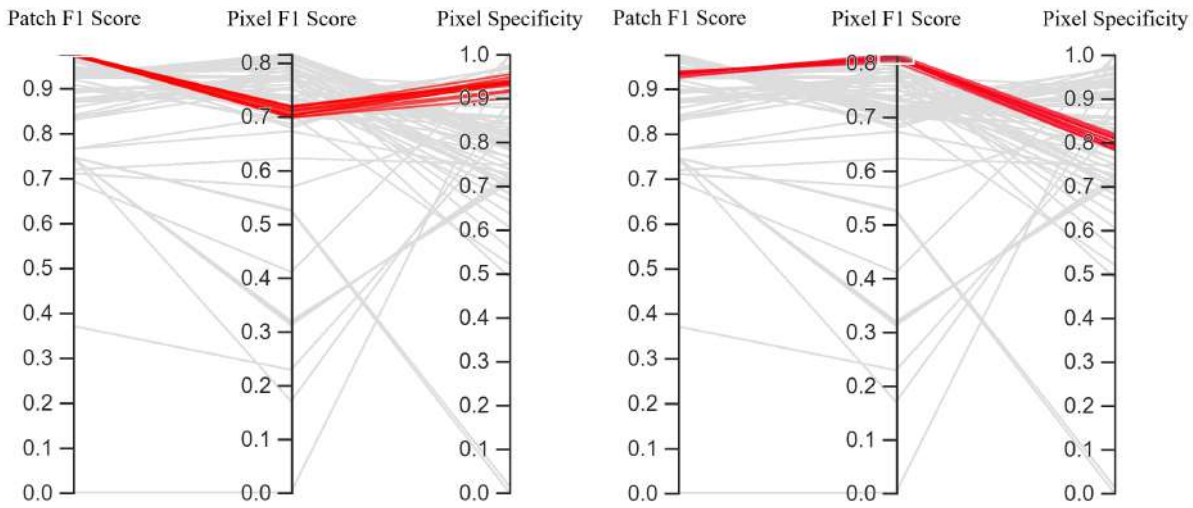


Figure 5.5: *Patch F1 score* versus patch size for different SwinMIL models after hyperparameter optimization. Each point marks a model with different hyperparameters. For almost every patch size there are models with high *patch F1 scores* > 0.9.

The *patch F1 score* in Figure 5.4b started at 0.831 and plateaus after 100K samples around 0.92 with some fluctuations afterwards. In contrast, the *pixel F1 score*, stays in a relatively narrow range around 0.64 – 0.67 from the first checkpoint on. This is lower than the *patch F1 score*, indicating that while the model is good at classifying patches, it struggles when it comes to classifying individual pixels within those patches. The *pixel specificity* increases throughout the training, indicating that the model is gradually getting better at correctly identifying *non-cancerous* pixels in healthy slides, i.e. making fewer false positives. The Hausdorff distance fluctuates throughout the training between 50 and 60. Some fluctuation is to be expected since it measures the spatial accuracy of the segmentation and depends on both the classification accuracy of the pixels and the spatial distribution of the errors.

Experiment 2: Finding Optimal Patch Size

The hyperparameter optimization revealed that there is no superior patch size for classifying patches with SwinMIL. Figure 5.5 contains all models from the hyperparameter optimization and shows the correlation between the *patch F1 score* and the patch size. For almost every patch size between 200×200 and 900×900, there are models with a high *patch F1 score* > 0.9. Therefore, the patch size does not play a critical role for the *patch F1 score* which is used to determine the best model. It was fixed to 400×400 for the following experiments and an additional hyperparameter optimization was performed.

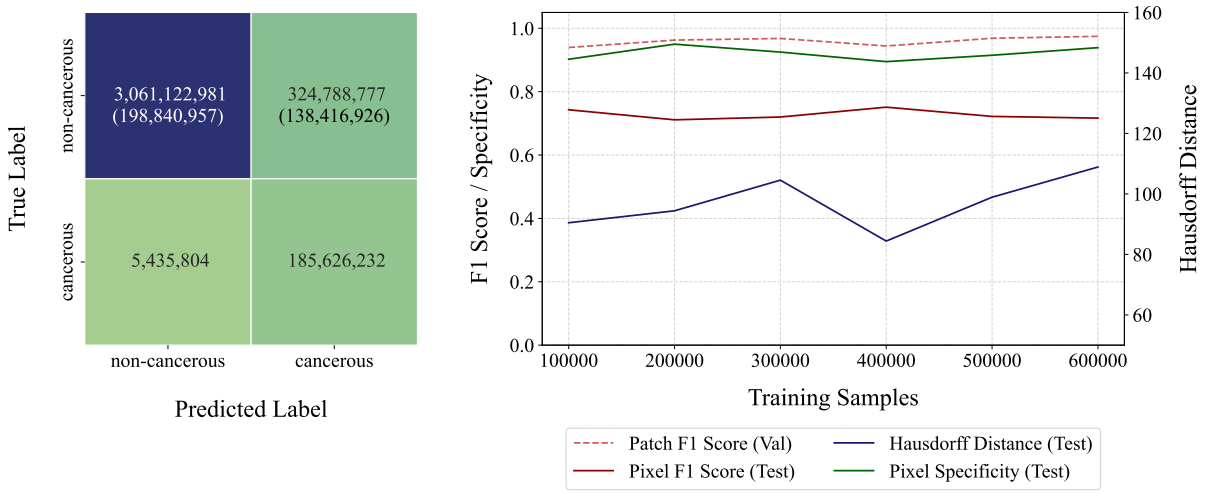


(a) Best *Patch F1 Scores* **(b)** Best *Pixel F1 Scores*

Figure 5.6: Correlation between metrics demonstrated on SwinMIL models from hyperparameter optimization. Each line represents a model with different hyperparameters and shows the corresponding *patch F1 score*, *pixel F1 score* and *pixel specificity* after the optimization with BOHB. In (a) and (b), 5% of the models with the best *patch F1 scores* and the best *pixel F1 scores* are marked in red, respectively.

Next, the correlation between *patch F1 score* on patch level and segmentation performance on pixel level was analyzed. Since the *patch F1 score* serves as the measure to select the best model, a high correlation between *patch F1 score* and the final segmentation performance on the test set is desirable. Therefore, a model with a high *patch F1 score* should also have a high *pixel F1 score*. The same should hold for the relationship between *patch F1 score* and *pixel specificity*. Figure 5.6 visualizes the correlation between these three metrics for all models from the hyperparameter optimization on patch size 400×400. It can be seen that the models with the highest *patch F1 score* do not have the highest *pixel F1 score*. There are other models with a lower *patch F1 score* that have a better *pixel F1 score*, that is, a better segmentation performance on the *cancerous* WSIs. The *pixel specificity* for the models with the best *patch F1 scores* is rather scattered and varies between 0.90 and 0.96. Figure 5.6b shows that the models with the highest *pixel F1 scores* have a relatively low *pixel specificity*, i.e. they produce many false positives in *healthy* slides. This suggests that there is a trade-off between the *pixel F1 score* and *pixel specificity*. This indicates that the model's ability to correctly classify patches cannot be directly translated into the model's ability to predict the class of individual pixels.

Furthermore, the performance of the best high-resolution model on patch size 400×400 was analyzed. The numerical results are shown in Figure 5.7. The model was trained with 600K patches and for every 100K patches the metrics shown in (b) were computed. It can be seen that the *patch F1 score* already has a relatively high value of 0.940 at the first checkpoint. It fluctuates slightly during training and reaches its highest value of 0.975 at the last checkpoint.



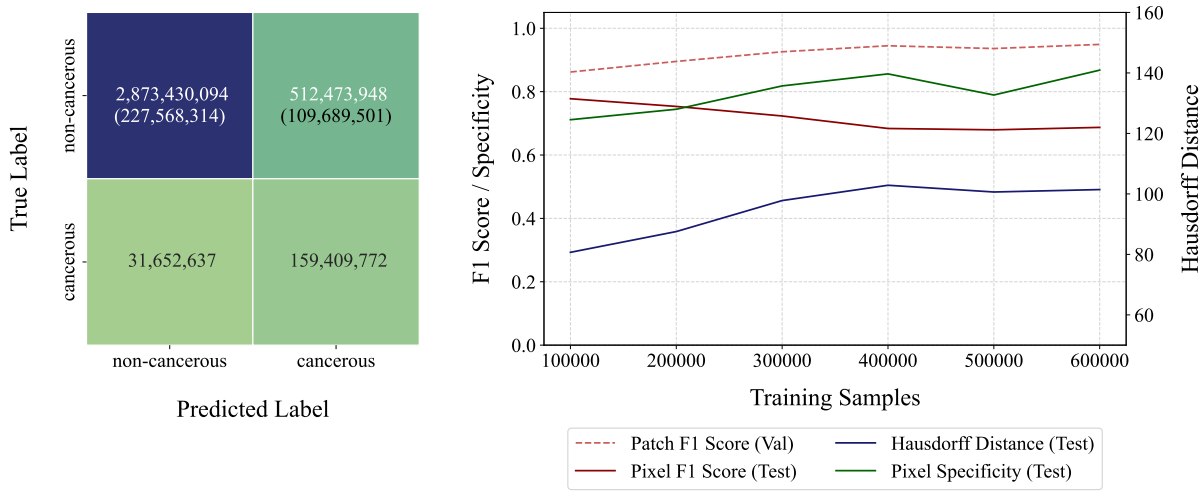
(a) Confusion Matrix (b) Metrics

Figure 5.7: Numerical results for the best high-resolution SwinMIL model on patch size 400 × 400. The confusion matrix in (a) shows the performance of the best checkpoint on the test set. The number in brackets denotes the amount of patches originating from *cancerous* WSIs. In (b), the evolution of the metrics during the training of the model is shown. The *pixel F1 score* on the test set was calculated for all checkpoints, but not used for model selection.

The *pixel F1 score* is significantly lower than the *patch F1 score* throughout the training. Furthermore, the *pixel F1 score* does not show a consistent upward trend. After an initial value of 0.743 at 100K samples, it fluctuates until it reaches its maximum of 0.751 at 400K samples and then drops back to 0.717 at 600K samples. The corresponding graph indicates that further training after 100K samples has no positive impact on the *pixel F1 score*. The same behavior can be observed for the *pixel specificity*. It starts with a value of 0.902 at 100K and reaches its highest value of 0.950 at 200K samples, after which further training seems to have no benefit. The graph of the Hausdorff distance shows fluctuations and reaches its lowest value of 84.4 at 400K samples. The value at 600K samples, which corresponds to the best *patch F1 score* is 108.9. The Hausdorff distance does not appear to be correlated with the other metrics.

Experiment 3: Improving Performance by Pretraining with DINO

The results in this experiment are compared to the best high-resolution model on patch size 400 × 400 from the previous Experiment 2. The best model from Experiment 2 is referred to as the reference model in the following. Pretraining the Swin Transformer backbone with DINO on the CancerScout-CRC dataset showed no improvement in performance compared to the reference model which was initialized with ImageNet weights. The numerical results of the best model pretrained with DINO are shown in Figure 5.8. The *patch F1 score* generally improves as the number of samples increases and reaches its highest value of 0.949 at the final checkpoint. The corresponding *pixel F1 score* of 0.687 is lower than for the reference model,



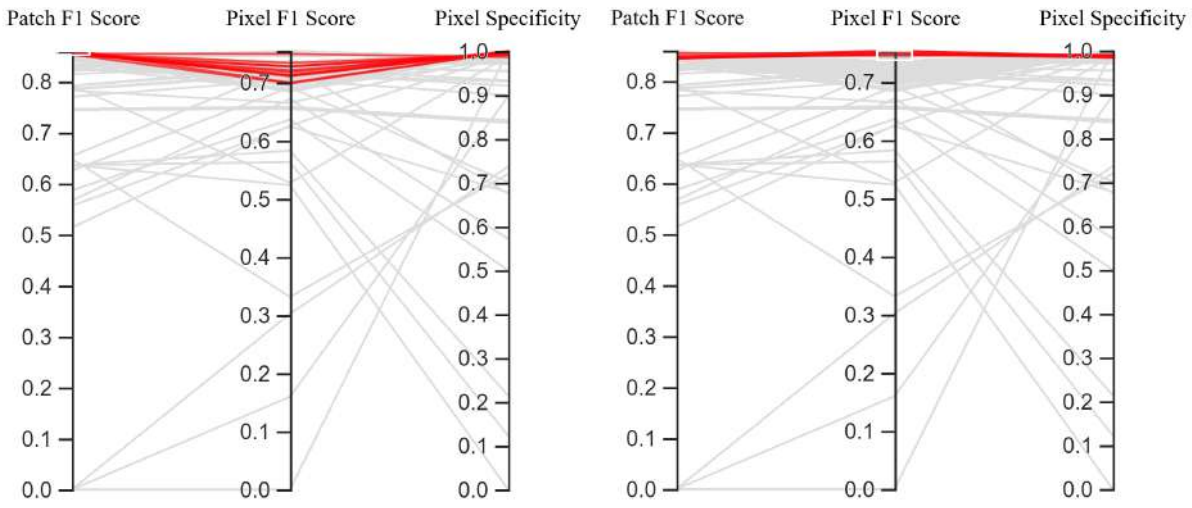
(a) Confusion Matrix (b) Metrics

Figure 5.8: Numerical results for the best high-resolution SwinMIL model pretrained with DINO. The confusion matrix in (a) shows the performance of the best checkpoint on the test set. The number in brackets denotes the amount of patches originating from *cancerous* WSIs. In (b), the evolution of the metrics during the training of the model is shown. The *pixel F1 score* on the test set was calculated for all checkpoints, but not used for model selection.

which is 0.717. Although the *pixel specificity* generally improves in this experiment as the number of samples increases, its value of 0.868 at the final checkpoint is still lower than the corresponding final *pixel specificity* of 0.939 from the reference model. It should also be noted that the *pixel specificity* of the reference model is much better across all checkpoints during training. The Hausdorff distance generally increases during training to a value of 101.4 at the last checkpoint. This is lower than the Hausdorff distance of 108.9 from the reference model. Nevertheless, the lower *pixel F1 score* and the *pixel specificity* suggest that DINO even has a negative impact on the final segmentation performance, if the best model is selected based on the *patch F1 score*.

Experiment 4: Improving Performance by Hard Negative Mining

The results of this experiment are compared to the high-resolution models on patch size 400×400 from Experiment 2. First, the correlation between the *patch F1 score* and the segmentation performance on pixel level was analyzed. Figure 5.9 shows the correlation between these three metrics for all models from the hyperparameter optimization. Compared to Experiment 2, the best 5% of models in (a) have a lower *patch F1 score*. However, the corresponding *pixel F1 scores* for these models are still all above 0.7, as in Experiment 2. Furthermore, it is noticeable that the models with the best *patch F1 scores* are now also the models with the best *pixel F1 scores*. Thus, there seems to be a better correlation between *patch F1 score* and segmentation performance on *cancerous* slides. The pixel specificity has improved significantly by training

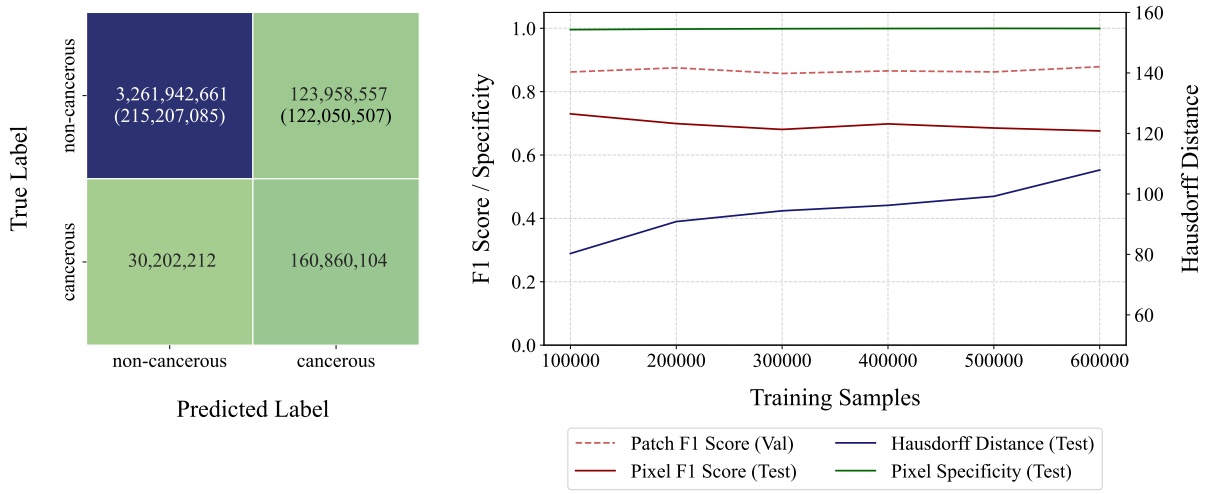


(a) Best *Patch F1 Scores* **(b)** Best *Pixel F1 Scores*

Figure 5.9: Correlation between metrics demonstrated on SwinMIL models trained with hard negatives. Each line represents a model with different hyperparameters and shows the corresponding *patch F1 score*, *pixel F1 score* and *pixel specificity* after the optimization with BOHB. In (a) and (b), 5% of the models with the best patch F1 scores and the best *pixel F1 scores* are marked in red, respectively.

with hard negatives and is always above 0.98 for the 5% of the models with the best *patch F1 scores*. This indicates that training with hard negatives has reduced the false positives in *healthy* slides. In addition, the *pixel specificity* now shows less variance between the models with the best *patch F1 scores*. Figure 5.9b shows the models with the best *pixel F1 scores*. The best *pixel F1 scores* are slightly lower than in Experiment 2. Here vary between 0.74 – 0.76, while in Experiment 2 they varied between 0.80 – 0.82. However, since the best model is selected based on the *patch F1 score*, this does not directly affect the performance of the best model. It is also worth noting that the models with the best *pixel F1 scores* also have relatively *high pixel specificity*, which was not the case in Experiment 2.

The numerical results of the best SwinMIL model in this experiment are shown in Figure 5.10 and are compared to the best model from Experiment 2 which will be referred to as the reference model in the following. The reference model outperforms the model in this experiment in terms of the *patch F1 score*. The reference model's *patch F1 scores* range from approximately 0.94 – 0.98 during training, which are significantly higher than the *patch F1 scores* of this model (0.86 – 0.88). This suggests that the reference model is more accurate in terms of patch-level classification. In terms of *pixel F1 score*, the performance of the two models is closer. The *pixel F1 scores* of the reference model range from about 0.71 – 0.75, while *pixel F1 scores* of the model in this experiment range from about 0.68 – 0.73. There's a major difference between the models in terms of *pixel specificity*. The model trained with hard negative examples in this experiment outperforms the reference model significantly in this aspect. The *pixel specificity* of



(a) Confusion Matrix (b) Metrics

Figure 5.10: Numerical results for the best high-resolution SwinMIL model trained on hard negatives. The confusion matrix in (a) shows the performance of the best checkpoint on the test set. The number in brackets denotes the amount of patches originating from *cancerous* WSIs. In (b), the evolution of the metrics during the training of the model is shown. The *pixel F1 score* on the test set was calculated for all checkpoints, but not used for model selection.

the model in this experiment ranges from 0.995 to 0.999, while the reference model's specificity is lower, ranging from 0.90 to 0.95. This implies that training with hard negative examples is very effective in reducing the false negatives in the *healthy* slides. The Hausdorff distance for the two models shows a similar trend of increasing over the course of the training. Both models' Hausdorff distances start at around 80 – 90 and increase to over 100.

5.2.2 Qualitative Results

This section presents and compares the qualitative segmentation masks of the best models from the four experiments. First, the results are illustrated on selected 400×400 patches. This includes examples where the high-resolution models demonstrated a successful segmentation of cancerous tissue as well as the limitations of the high-resolution SwinMIL models, i.e. less successful segmentation results. In addition, the qualitative effects of hard negative mining will be investigated in more detail at the slide level.

Comparison of Segmentation Masks on Selected Patches

To select the successful segmentation examples, the high-resolution model from Experiment 2 on patch size of 400×400 was used. It is referred to as the reference model in the following. Specifically, patches with a high *pixel F1 score* were selected, where prediction and ground truth matched well. Figure 5.11 shows the prediction masks of the best models from the four

experiments on six different 400×400 WSI patches. All patches were extracted from regions annotated as *cancerous* from *cancerous* WSIs. The first three patches in Figure 5.11 were selected from *cnew* slides and the following three patches from *cold* slides. The high-resolution models in (b) – (d) do not show significant visual differences in the quality of their segmentation masks. All high-resolution models distinguish cancerous tissue from surrounding healthy tissue similarly well in these examples. Medium-sized regions of healthy tissue surrounded by cancerous tissue also appear to be reliably excluded. However, if the healthy tissue regions in between cancerous tissue are too small, the high-resolution models have difficulty excluding it properly. The low-resolution SwinMIL model in Figure 5.11a, on the other hand, provides significantly worse segmentation masks. It detects cancerous tissue in most cases but suffers from oversegmentation and cannot properly segment the cancerous tissue structures. In one case (the last row in Figure 5.11a) the low-resolution SwinMIL model does not detect the cancerous tissue at all. However, the low-resolution model appears to have sufficient resolution to correctly represent most of the details of the ground truth masks for the selected patches. The ground truth masks rarely include morphological tissue structures (e.g. non-epithelial tissue gaps or fine tubular epithelium) smaller than 12 mpp.

The reference model from Experiment 2 was also used to select the examples where the high-resolution segmentation was unsuccessful. Figure 5.12 shows six 400×400 patches where the reference model failed to correctly segment the cancerous tissue and compares them with the best models from the other three experiments. The patches in the first three rows in Figure 5.12 were selected from *cancerous* slides and the patches in the following three rows from *healthy* slides. It can be seen that the reference model in (b) significantly oversegments the cancerous tissue in the *cancerous* patches. This problem is even more pronounced for the low-resolution SwinMIL model in (a), which classifies almost all pixels within the patches as *cancerous*. The DINO model in (c) also produces many false positives in *cancerous* patches, although slightly fewer than the reference model. In contrast, the SwinMIL model in (d), which was trained on negatively mined samples, can significantly reduce false positives and suffers less from oversegmentation than the reference model. However, it also shows slight undersegmentation in some areas of the *cancerous* patches (blue regions in the overlay). Comparing the prediction masks on the *healthy* patches (last three rows in Figure 5.12), the difference between the model trained with negatively mined samples and the other models becomes even more pronounced. SwinMIL with negative mining recognizes significantly less healthy tissue as cancerous. While the SwinMIL model with DINO in (c) performs similarly to the reference model for *cancerous* patches (it even performs better for the specifically selected *cancerous* patches in Figure 5.12), it clearly does produce more false positives for the *healthy* patches. This also aligns with the numerical results, where the *pixel specificity* of the DINO model is significantly lower compared to the reference model. The low-resolution SwinMIL model in (a) detects almost all

tissue as *cancerous* within the *healthy* patches. In one case (the last row of Figure 5.12a), even background pixels are classified as *cancerous*.

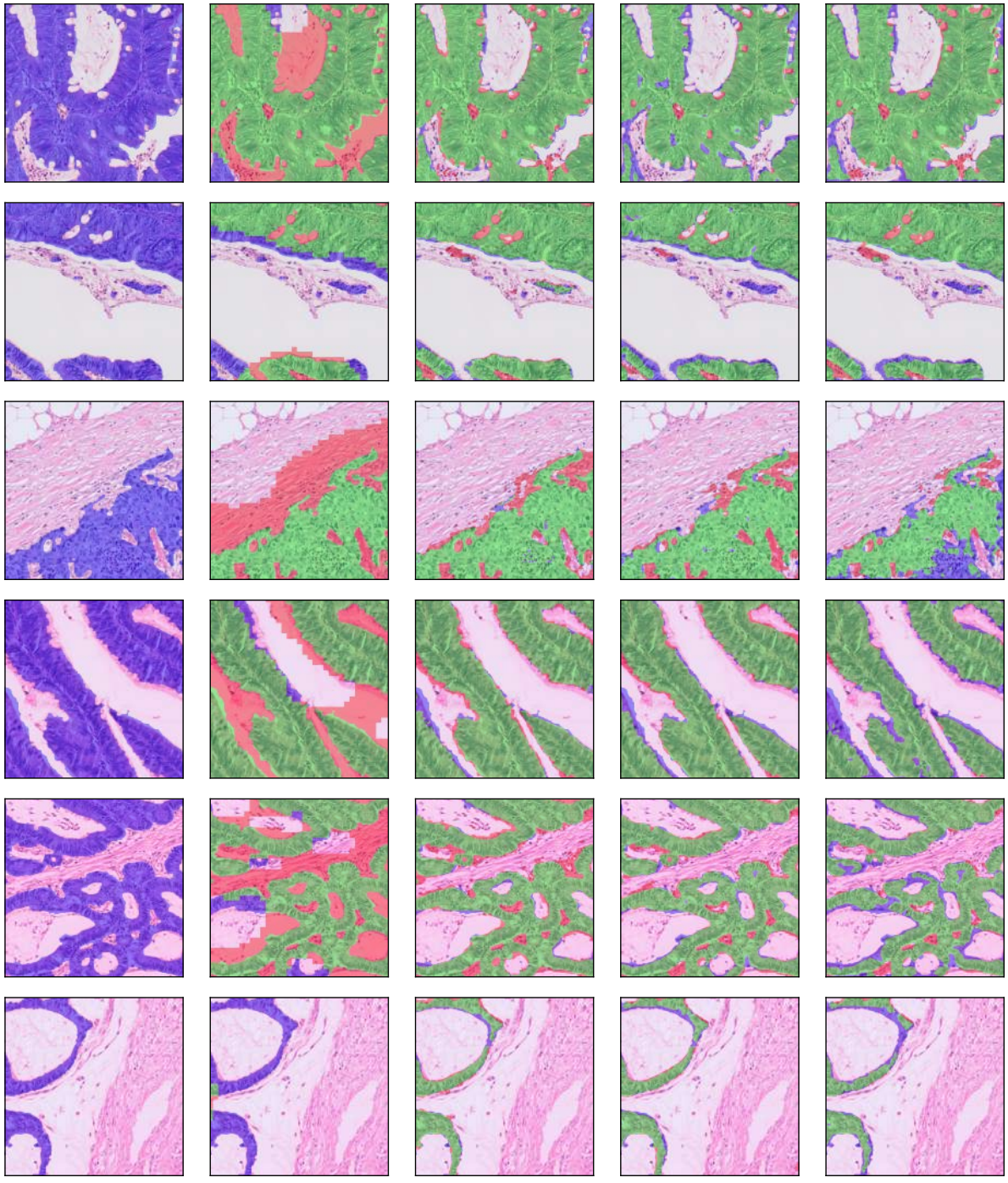


Figure 5.11: Successful segmentation examples on 400×400 patches. The first column shows the generated ground truth (GT). The other columns show an overlay of the model predictions from the different experiments with the ground truth mask. Regions where the model correctly detects cancer tissue (true positives) are colored green. False positive predictions are colored red and false negatives are colored blue. (a) shows the prediction of the low-resolution SwinMIL model from Experiment 1, (b) shows the prediction of the reference model from Experiment 2, and (c) and (d) show the predictions of the high-resolution models from Experiments 3 and 4, trained with DINO and hard negative mining, respectively.

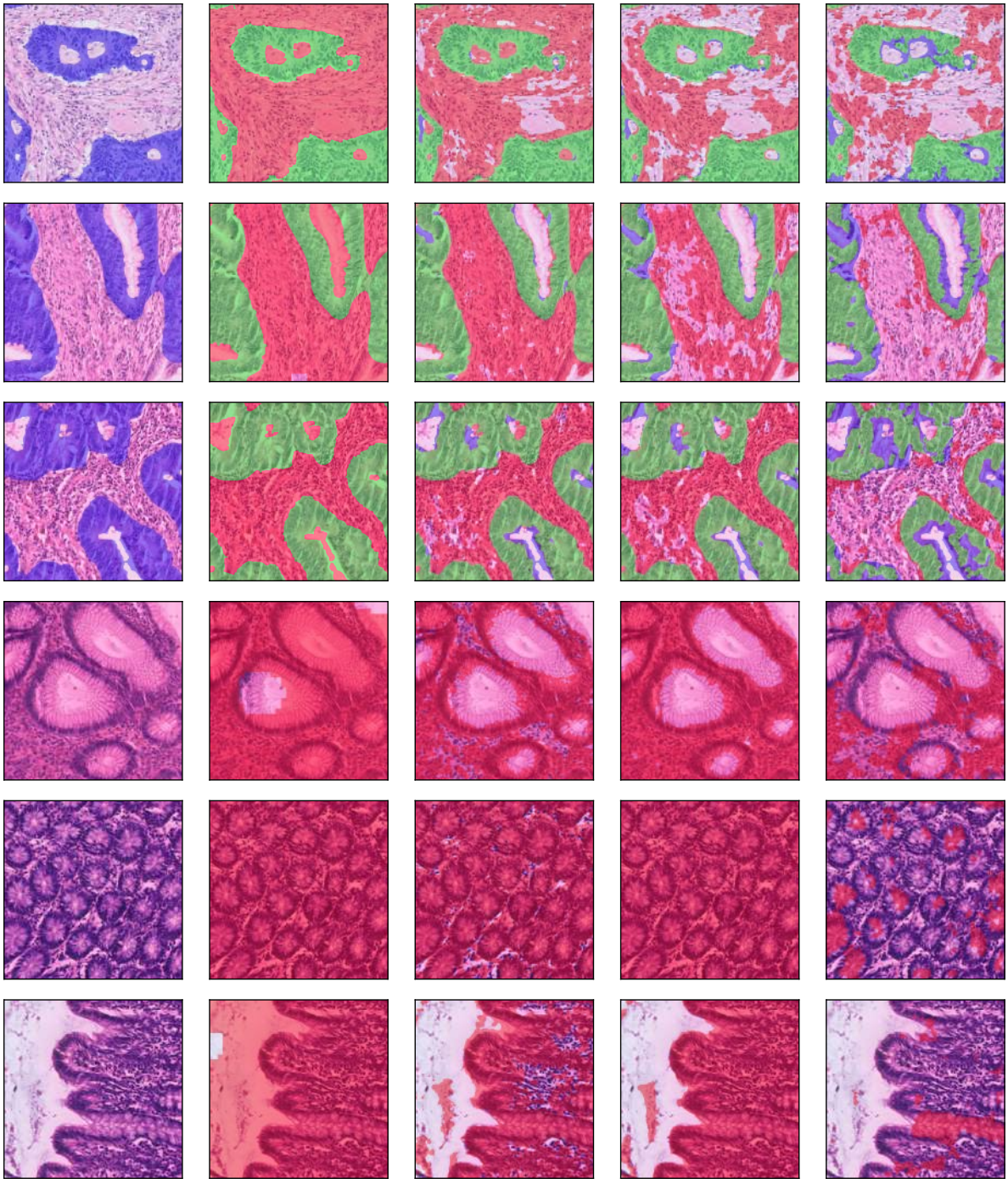
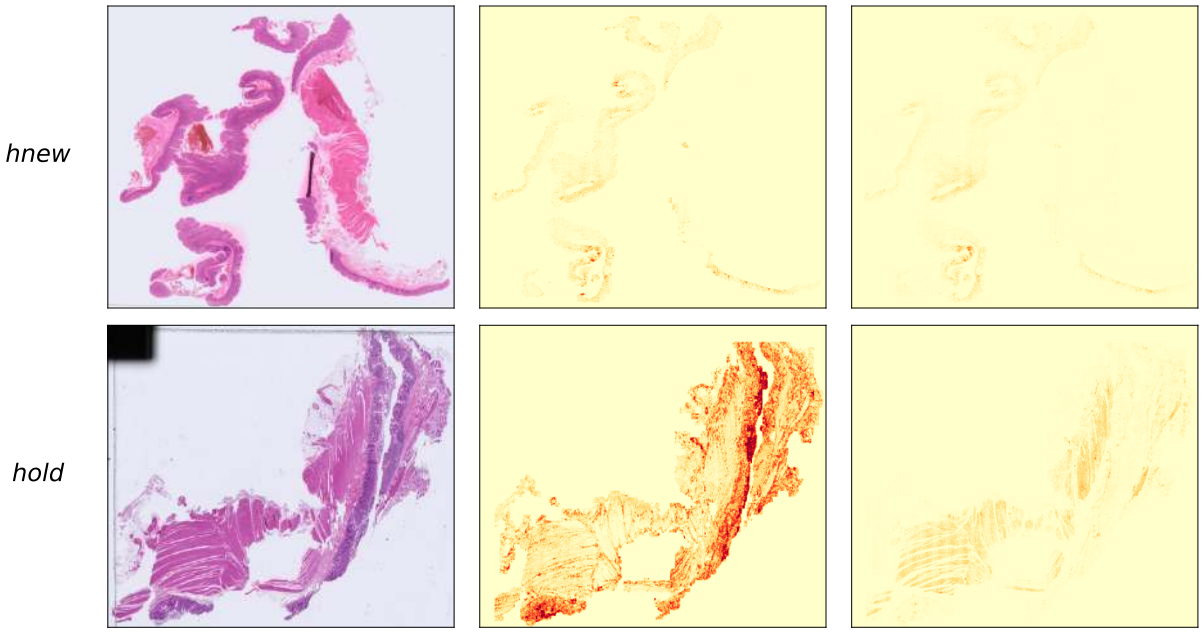


Figure 5.12: Unsuccessful segmentation examples on 400 × 400 patches. The first column shows the generated ground truth (GT). The other columns show an overlay of the model predictions from the different experiments with the ground truth mask. Regions where the model correctly detects cancer tissue (true positives) are colored green. False positive predictions are colored red and false negatives are colored blue. (a) shows the prediction of the low-resolution SwinMIL model from Experiment 1, (b) shows the prediction of the reference model from Experiment 2, and (c) and (d) show the predictions of the high-resolution models from Experiments 3 and 4, trained with DINO and hard negative mining, respectively.



(a) Healthy WSI (b) Prediction Reference (c) Prediction Negative Mining

Figure 5.13: Comparison of predictions on two *healthy* WSIs between the reference model from Experiment 2 and the model trained with hard negative examples in Experiment 4. The prediction masks were obtained using the best models from both experiments.

Visualizing Effect of Negative Mining on WSI Level

To better illustrate the effect of hard negative mining on SwinMIL, the prediction masks were also compared with the reference model on slide level. Figure 5.13 shows the predictions of both models on one *hnew* and one *hold* WSI from the test set. Here, the same observations can be made as already in Figure 5.12 for individual patches: SwinMIL produces fewer false positives after training with hard negative samples. This effect is particularly evident for the *hold* WSI in Figure 5.13, where the reference model recognizes a particularly high amount of cancerous tissue. It is also noteworthy that the most false positives in *healthy* slides seem to be in epithelial tissue. This can be observed for all *healthy* slides of the test set. A more comprehensive comparison of the prediction masks of both models on all *healthy* WSIs from the test set can be found in Figure D.2 and Figure D.3. Overall, the qualitative results align with the numerical results from the previous section, where the best negative mining model achieves a significantly higher *pixel specificity* of 0.999 compared to the *pixel specificity* of 0.946 for the best reference model.

5.3 Analyzing Output of Different SwinMIL Stages

This section presents an analysis of the output generated by different stages of the SwinMIL model. This aims to explain the behavior and the limitations of SwinMIL in more detail to

identify potential areas of improvement. The best model on patch size 400×400 from Experiment 2 was used to analyze the predictions of the different stages on the test set. Table 5.2 compares the *pixel F1 score* and the Hausdorff distance between the three SwinMIL stages and the Fusion stage on the *cancerous* WSIs. It shows that the first SwinMIL stage provides both the highest *pixel F1 score* and the lowest Hausdorff distance for each of the four *cancerous* WSIs. In particular, Stage 1 shows a better performance than the Fusion Stage, which provides the final segmentation mask in the standard SwinMIL model. Since the Fusion Stage is a weighted average of the first three stages, this suggests that Stage 2 and Stage 3 have a negative effect on the final segmentation performance for *cancerous* WSIs. A closer look shows that the performance for all *cancerous* WSIs gradually decreases from Stage 1 to Stage 3 with Stage 3 providing the worst segmentation masks according to the *pixel F1 score* and the Hausdorff distance. The visualization of the segmentation masks in Figure 5.14 provides a more accurate picture of the performance degradation across the stages. It shows that Stage 1 provides the most detailed and accurate segmentation mask and that the oversegmentation increases in Stage 2 and Stage 3. This is reasonable as the resolution of the feature maps in SwinMIL decreases after each stage due to downsampling. Consequently, the segmentation mask loses detail when it is upsampled to the original size of the WSI. A more comprehensive comparison of the stages for all *cancerous* WSIs in the test set can be found in Figure E.1.

Table 5.3 compares the *pixel specificity* between all stages on the *healthy* WSIs from the test set. Here, an opposite pattern emerges: The segmentation mask from Stage 1 always provides the lowest *pixel specificity*, i.e. it produces the most false-positive pixel predictions. The segmentation mask from Stage 3, on the other hand, delivers the best result on the *healthy* WSIs according to *pixel specificity* and makes the fewest false positive predictions. This is confirmed when visualizing the prediction masks for all stages in Figure 5.15. While Stage 1 performed best for *cancerous* slides and Stage 3 performed worst, the reverse is true for *healthy* WSIs. Another observation in the *healthy* slides is that the first stage seems to recognize mainly the epithelial tissue as *cancerous*. A more comprehensive comparison of the stages for all *hnew* and *hold* WSIs in the test set can be found in Figure E.2 and Figure E.3, respectively.

Slide	*Pixel F1 Score*				*Hausdorff Distance*			
	Stage 1	*Stage 2*	*Stage 3*	*Fusion*	*Stage 1*	*Stage 2*	*Stage 3*	*Fusion*
cnew 1	**0.677**	0.610	0.549	0.612	**131.6**	163.9	189.4	165.9
cnew 2	**0.786**	0.679	0.637	0.695	**111.8**	151.3	196.1	159.8
cold 1	**0.750**	0.686	0.679	0.727	**114.2**	130.0	184.7	132.0
cold 2	**0.638**	0.499	0.411	0.516	**146.7**	195.4	237.8	203.1

Table 5.2: Comparison of the *pixel F1 Score* and Hausdorff distance across all SwinMIL stages for the *cancerous* WSIs in the test set using the best reference model. The best value for each WSI is shown in bold.

Slide	*Stage 1*	*Stage 2*	*Stage 3*	*Fusion*
hnew 1	0.946	0.998	**1.000**	0.998
hnew 2	0.932	0.993	**0.999**	0.996
hnew 3	0.958	0.995	**0.999**	0.996
hnew 4	0.846	0.944	**0.984**	0.957
hnew 5	0.960	0.996	**0.999**	0.998
hold 1	0.812	0.932	**0.969**	0.940
hold 2	0.742	0.867	**0.962**	0.901
hold 3	0.747	0.922	**0.982**	0.941
hold 4	0.583	0.830	**0.977**	0.872
hold 5	0.671	0.915	**0.983**	0.936

Table 5.3: Comparison of the *pixel specificity* across all SwinMIL stages for the *healthy* WSIs in the test set using the best reference model. The best value for each WSI is shown in bold.

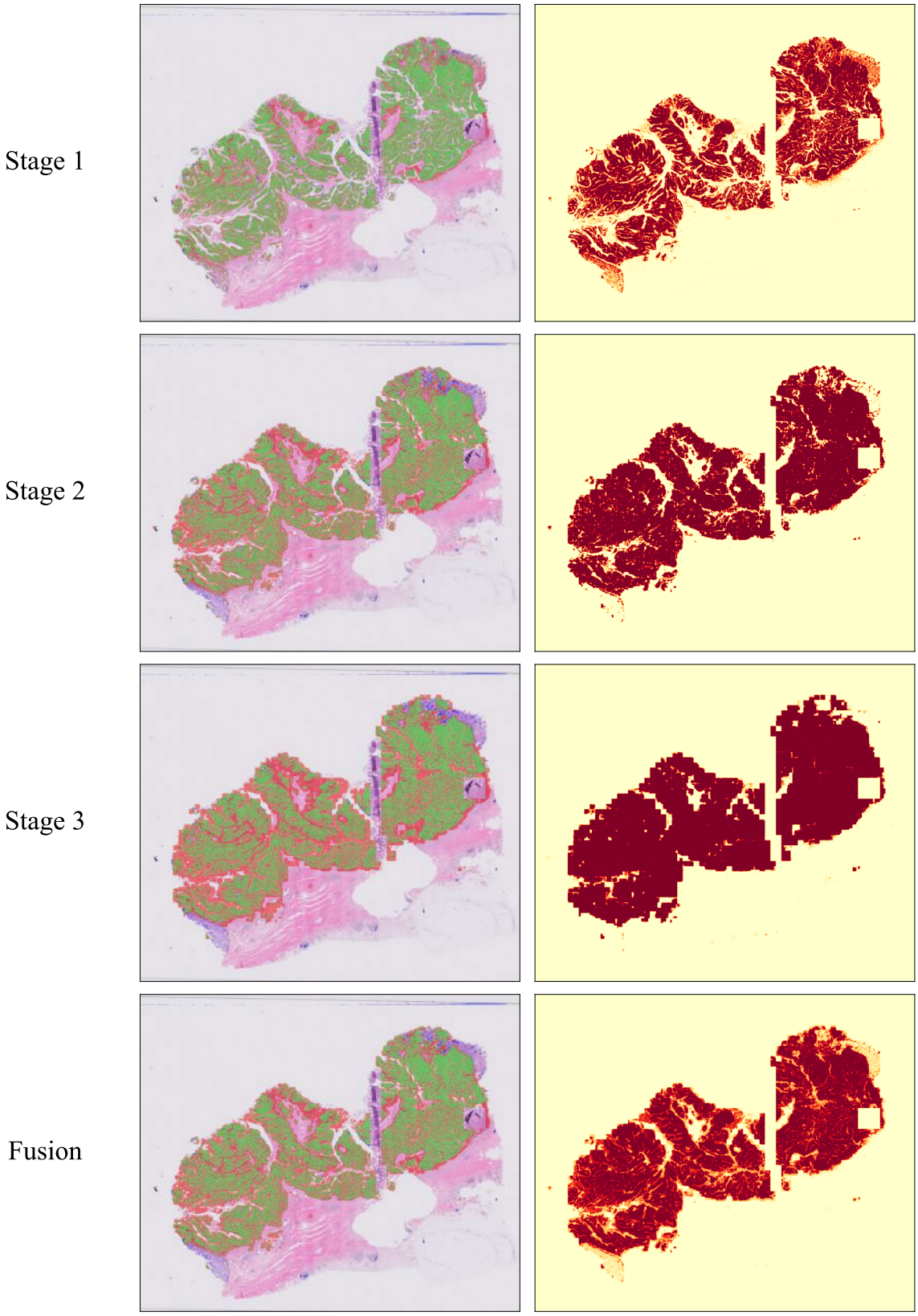


Figure 5.14: Comparison of the segmentation masks across all SwinMIL stages on a *cancerous* WSI from the test set. An overlay of the stage predictions with the ground truth mask is shown in (a). True positive predictions are colored green, false positives red, and false negatives blue. A raw heatmap of the prediction values is presented in (b).

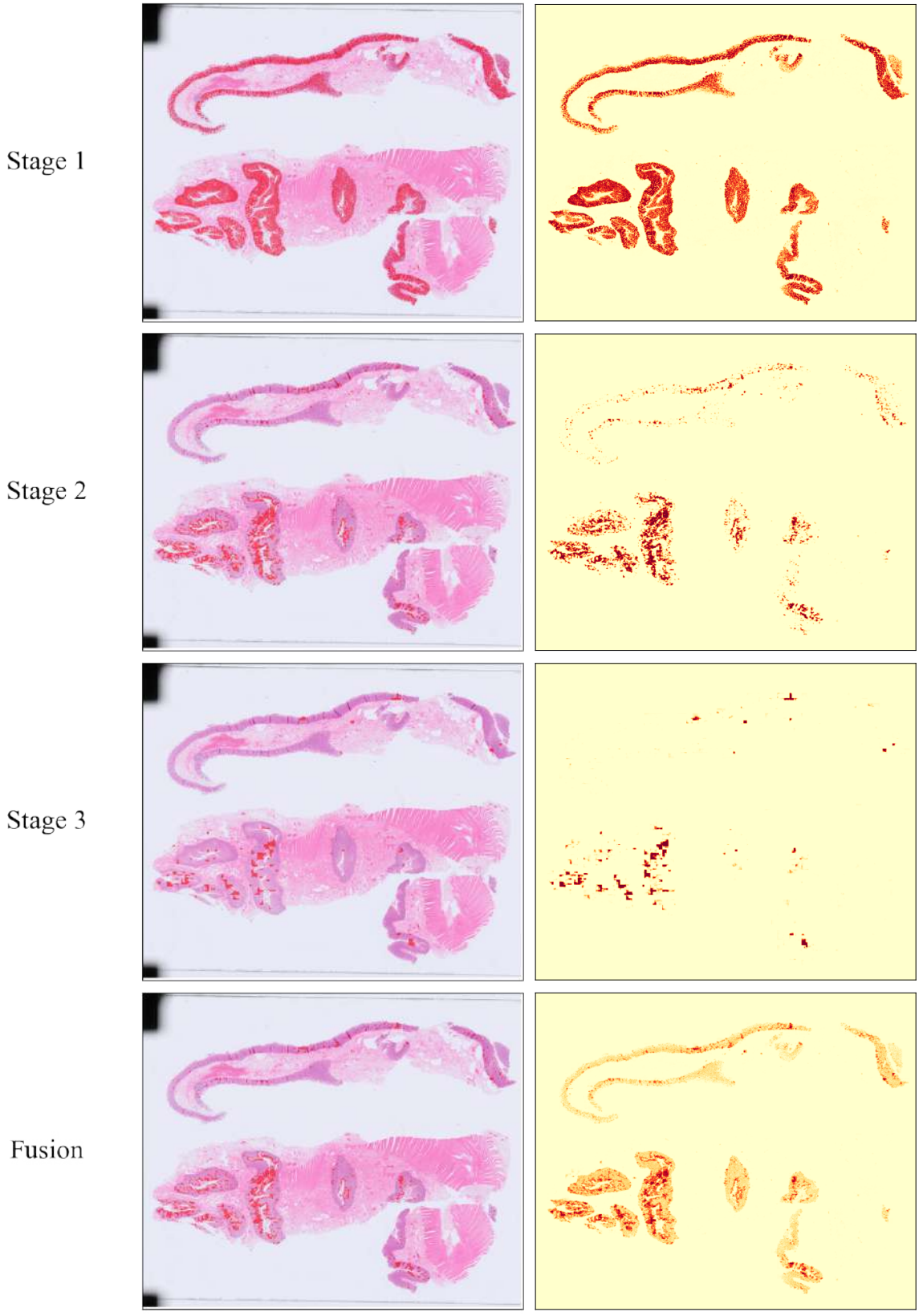


Figure 5.15: Comparison of the segmentation masks across all SwinMIL stages on a *healthy* WSI from the test set. An overlay of the stage predictions with the WSI is illustrated in (a) and a raw heatmap of the prediction values in (b).

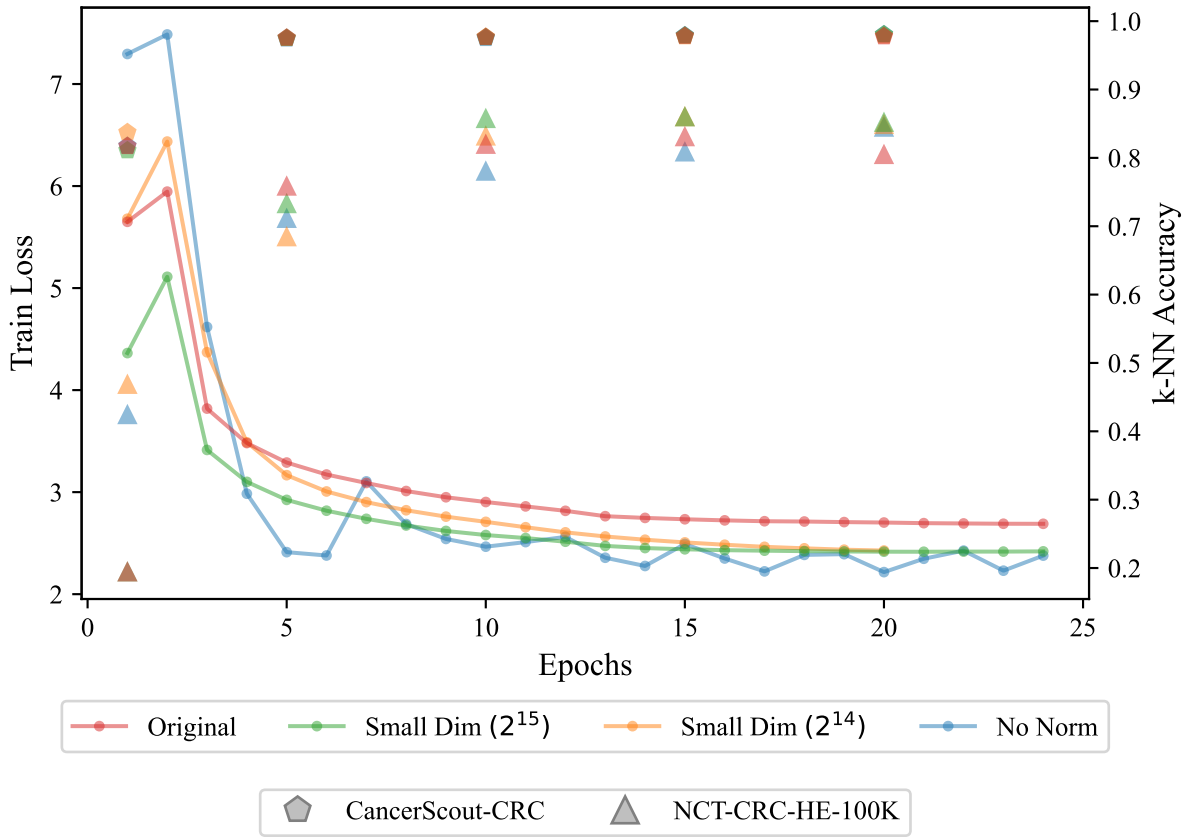
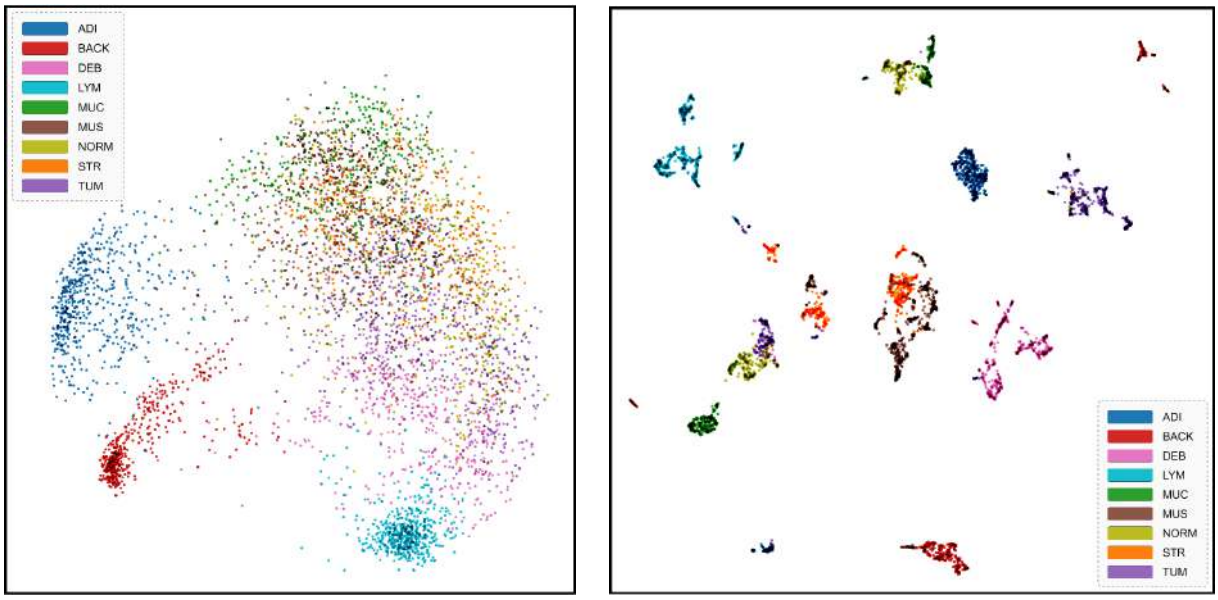


Figure 5.16: Comparison of the numerical results between the four different DINO pretraining configurations. The average training loss was calculated after each epoch while the k-NN classification tasks on the CancerScout-CRC and NCT-CRC-HE-100K datasets were performed every 5 epochs for each configuration. The classification accuracy is reported for both classification tasks.

5.4 Self-Supervised Pretraining with DINO

Figure 5.16 shows the comparison of the numerical results of the four different pretraining configurations. It can be seen that each model converges regardless of configuration. However, the configurations where the dimensionality of the output vectors was decreased as well as the configuration where the weight normalization was removed, converge to a lower final loss than the original default configuration. This implies that these configurations are learning the training data more effectively. The model with removed weight normalization exhibits unstable behavior. Considering the k-NN classification on the CancerScout-CRC dataset, the accuracy reaches a value of > 0.97 at epoch 5 for all configurations and varies between 0.97 – 0.98 for the rest of the training. Therefore, further training from epoch 5 onwards does not improve the performance of any configuration on the binary classification problem (*cancerous* / *non-cancerous*) of the CancerScout-CRC dataset. When looking at the k-NN classification accuracy on the NCT-CRC-HE-100K dataset, more differences between the configurations be-



(a) PCA **(b)** UMAP

Figure 5.17: Visualization of the feature vectors after dimensionality reduction extracted from 5000 WSI patches. The outcome of PCA is shown in (a). The two axes represent the first two principal components. The outcome of UMAP is shown in (b).

come apparent. Here, the model with output dimensionality reduced to 2^{15} stands out with the highest k-NN accuracy of 0.86 at epoch 15. This result suggests that simplifying the feature space can potentially improve model performance when training with DINO. It should also be noted that for each configuration, the k-NN classification accuracy on the NCT-CRC-HE-100K dataset increases up to epoch 15. In epoch 20, however, it starts to decrease again for almost all configurations. The exception is the configuration where normalization was omitted, which classification accuracy still seems to increase after epoch 15.

For the best pretraining configuration *Small Dim* 2^{15}, the features computed on the patches of the NCT-CRC-HE-100K dataset were visualized. Figure 5.17 shows the two-dimensional feature vectors after dimensionality reduction with PCA and UMAP. After applying PCA, three clusters - adipose (ADI), background (BACK), and lymphocytes (LYM) - distinctly stand out from the rest of the datapoints. Visually, the intra-class distance for these three classes appears smaller than for the other classes, indicating a higher degree of similarity between their samples or a lower amount of variance. The samples within the other classes are more dispersed and overlap strongly with the datapoints of neighboring classes (small inter-class distance). Therefore, not all classes can be adequately separated into different clusters after dimensional reduction with linear PCA. This includes in particular samples of the class colorectal adenocarcinoma epithelium (TUM). After the dimensionality reduction with UMAP, on the other hand, the sample features tend to have a more localized structure for each class. Furthermore, UMAP produces a more defined separation between the different classes. However, some classes, such

as cancer-associated stroma (STR), colorectal adenocarcinoma epithelium (TUM), or normal colon mucosa (NORM), are separated into multiple clusters that are further apart from each other in the two-dimensional representation space. There is also some overlap between different clusters, although not to the extent of PCA. Overall, UMAP manages to handle the issue of wide scatter and strong overlap of datapoints from neighboring classes. The visualization confirms that the pretrained Swin Transformer can produce useful features for histopathological image data.

6 Discussion

This chapter summarizes and interprets the general results of the high-resolution WSSS with SwinMIL. Overall, SwinMIL showed promising potential on high-resolution WSIs, achieving a high level of detail and converging quickly when initialized with ImageNet weights. The initial high-resolution model from Experiment 2 produced a considerable amount of false positives, especially for the epithelial tissue in *healthy* WSIs. In addition, the models with the highest *patch F1 score* did not have the highest *pixel F1 score*, i.e. the correlation between patch-level and pixel-level performance scores was not adequate. However, both problems were significantly mitigated by the application of hard negative mining in Experiment 4. Nevertheless, the qualitative segmentation masks showed, that SwinMIL is still prone to oversegmentation in the presence of cancerous tissue. A closer examination of the individual SwinMIL stages showed that this is mainly caused by Stage 3 and partly by Stage 2. Furthermore, the qualitative comparison with the low-resolution segmentation masks from Experiment 1 suggests that a resolution of less than 1 mpp is probably sufficient to segment most structures or details of cancerous epithelial tissue precisely. Contrary to expectations, pretraining the Swin Transformer backbone with DINO in Experiment 3 did not improve the WSSS performance with SwinMIL. The following sections describe other observations made during the individual experiments and provide possible explanations.

6.1 Low-Resolution SwinMIL Implementation

The low-resolution SwinMIL model in Experiment 1 suffers significantly more from oversegmentation than the high-resolution models, which is reflected in the lower *pixel F1 score*. One reason for this could be that the downscaling process causes the loss of valuable information that is important for accurately distinguishing between different tissue structures. The reduction in resolution may blur the boundaries between distinct regions, causing the model to incorrectly classify background or other non-epithelial tissue as *cancerous*. However, neglecting the oversegmentation, the qualitative results show that despite downscaling, the generated segmentation masks with 12 mpp have sufficient resolution to capture most ground truth structures correctly. Therefore, the benefit of high resolution for the segmentation of tumorous epithelial tissue appears to be negligible in this case, since tiny morphological structures (e.g., below 5 mpp) are rare. The advantage of high-resolution segmentation is likely to be more significant when it comes to segmenting tiny tissue structures such as cell nuclei. In this case, the ground truth masks would also need to represent these fine structures in order properly to evaluate the high-resolution prediction masks. For example, the ground truth masks generated in this thesis do not capture tiny details such as very small background regions surrounded by epithelial tissue.

6.2 High-Resolution SwinMIL Implementation

The results of Experiments 2 and 4 showed that the SwinMIL model converges quite fast on high-resolution patches when pretrained on ImageNet. This could be seen in the high *patch F1 score* and the high pixel metrics already at the beginning of the training. Further training of the SwinMIL model after 100K samples does not appear beneficial, as it did not significantly improve the *pixel F1 score* or the *pixel specificity* in these two experiments. Another observation is that the reference model in Experiment 2 produced many false positives in the *healthy* slides. One reason could be that the reference model was trained on healthy patches extracted exclusively from *cancerous* slides. Although these patches contained healthy tissue, their proximity to cancerous tissue may have altered their properties, making them different from healthy tissue found in entirely *healthy* slides. Therefore, the training data may not have represented the full variety of healthy tissue, resulting in worse performance on *healthy* slides. Furthermore, the reference model made significantly more false positives in the *hold* WSIs than in the *hnew* WSIs. This can be explained by the nature of the *hold* slides, which have different color characteristics and a higher color variability compared to the *hnew* slides. The increased variability could have made it more difficult for the model to learn the relevant features to distinguish cancerous from non-cancerous tissue in *hold* slides. Overall, SwinMIL is generally able to achieve a high level of detail at high resolution and detect tissue boundaries much better than the low-resolution implementation in Experiment 1, which suffers from oversegmentation.

6.3 Importance of Stage Weights in SwinMIL

The analysis of the individual SwinMIL stages showed that their weighting plays an essential role in balancing high- and low-level information in the final segmentation mask. Stage 1 primarily carries intricate, low-level details about the tissue structure and is responsible for most of the detail in the final segmentation mask. In particular, it seems to be effective in recognizing and segmenting epithelial cells. However, it is unable to differentiate between cancerous and non-cancerous epithelial tissue, probably because the features do not incorporate sufficient spatial context. The lower-resolved Stage 3, on the other hand, proved to be detrimental to the level of detail in the final segmentation map. In turn, it can more accurately identify whether the tissue is cancerous, which could be observed in the qualitative results of *healthy* WSIs. Therefore, there is a trade-off between achieving a high level of detail in the segmentation mask and producing a few false positives when choosing the weight for each stage.

The numerical results indicate that the default stage weights of SwinMIL are not optimal for the high-resolution segmentation of the CancerScout-CRC dataset. The fusion of the segmentation masks does not provide a direct benefit for *cancerous* or *healthy* WSIs. In fact, it has a rather negative effect. For *cancerous* slides, Stage 1 consistently outperformed the Fusion stage in

terms of *pixel F1 score* and Hausdorff distance. Similarly, Stage 3 always outperformed the fusion stage for *healthy* slides. This partially contradicts the results from the original paper, where for *cancerous* patches, the fused segmentation mask's *pixel F1 score* surpassed all individual stages. A possible explanation for this discrepancy could be the weight values assigned to each stage. The default weights for the three stages in the original work may not be optimal for the setup in this thesis, where SwinMIL is applied to high-resolution patches from a dataset with different characteristics. However, selecting the appropriate weights is not a straightforward task due to the inherent trade-off between high- and low-resolution stages described in the previous paragraph. Moreover, it is unclear how to determine the stage weights exactly. If the decision is based on pixel-level segmentation performance, this would inevitably use data from the pixel-level ground truth to optimize the SwinMIL algorithm, which contradicts the basic assumption of WSSS. Thus, one idea would be to make the stage weights learnable. The authors of the original work did not clarify how they determined the stage weights.

6.4 Increasing Trend of Hausdorff Distance

During the training of the high-resolution SwinMIL models in Experiments 2 – 4, the Hausdorff distance shows an increasing trend. A possible explanation for this behavior could be that as training progresses, the model becomes more confident in recognizing cancerous tissue, and therefore makes more positive predictions in the segmentation masks of Stage 3. Because these segmentation masks are low resolution, they cannot precisely delineate the tissue boundaries. However, given the fixed weight of 0.3 for Stage 3, it does consistently contribute to the final segmentation map. As a result, it may obscure the high-resolution tissue boundaries from Stage 1, which would lead to a decrease in the Hausdorff distance.

6.5 Effect of DINO Pretraining

It was observed that a supervised pretraining of the Swin Transformer backbone on ImageNet benefits the WSSS task with SwinMIL more than the self-supervised pretraining with DINO on histopathological images. This is an unexpected outcome for two reasons. First, it was hypothesized that a pretraining on histopathological images should be superior to a pretraining on natural images since similar histopathology images are also used in the WSSS downstream task with SwinMIL. Second, the authors of the original paper showed that the features derived from the self-supervised DINO pretraining contain information about the scene layout of the input images. This property of DINO was expected to benefit the semantic segmentation more than a supervised pretraining on ImageNet. A possible explanation for this unexpected result may lie in the different neural network architectures used. In the original paper, a Vision Transformer is pretrained with DINO while a Swin Transformer is employed in this thesis. The Vision Transformer has a more holistic understanding of the image content since the self-attention is

computed globally. This can lead to features that contain more complete information about the semantic segmentation of objects in the image. The Swin Transformer, on the other hand, computes the self-attention in localized windows and downsamples the feature maps during processing. As a result, the features after DINO pretraining may contain less information about the exact boundaries and extent of the objects. It was not possible to verify whether the features of the Swin Transformer contain information about the actual semantic segmentation of the images after DINO pretraining, because there is no mechanism to visualize the self-attention maps in the same way as for the Vision Transformer in the original paper. This is mainly because the Swin Transformer architecture does not compute the self-attention globally and does not use neutral [*class*] tokens. In fact, the work of Chen et al. [15] has already shown in a different setting that a Vision Transformer pretrained with DINO can capture useful features for the semantic segmentation of histopathological images.

6.6 Effect of Hard Negative Mining

Including the negatively mined examples in the training set not only reduced the false positives in *healthy* WSIs, but also improved the correlation between *patch F1 score* and the *pixel F1 score*. This suggests, that the selection of training data plays a critical role in the overall performance of the SwinMIL model. The reference model from Experiment 2 produced false positives in healthy slides mainly in epithelial tissue. Although the *cancerous* and *non-cancerous* patches were balanced in the training set of the reference model, it is possible that healthy epithelial tissue was underrepresented in the *non-cancerous* patches. One reason for this may be that the training was performed only on *cancerous* WSIs which are generally less likely to contain *healthy* epithelial tissue compared to *healthy* WSIs. Another reason for the underrepresentation of healthy epithelial tissue could be that the pathologists may have introduced an annotation bias when drawing the coarse regional annotations. It is conceivable that pathologists may have primarily annotated tissue as *non-cancerous* that was particularly easy to identify as *non-cancerous*. For example, this could be fat or muscle tissue, which by definition cannot be affected by adenocarcinoma and is visually very different from epithelial tissue. Therefore, the training dataset of the reference model may have contained few patches of non-cancerous epithelial tissue, which may have affected the ability of the model to discriminate between cancerous and non-cancerous epithelial tissue. This would be consistent with the finding that the reference model produced mainly false positives for epithelial tissue in *healthy* WSIs. Overall, the *pixel F1 score* of the best model was slightly lower after training with negatively mined examples. However, it remains competitive with the reference model. The significant improvements in *pixel specificity* clearly outweigh the disadvantage of the slightly lower *pixel F1 score*.

7 Outlook

The conducted experiments demonstrated that SwinMIL, when initialized with ImageNet weights, is effective in performing high-resolution WSSS on diverse WSIs and that hard negative mining can significantly reduce the false positives in this context. However, several challenges and opportunities for future work remain.

As noted in the discussion, the variation in color characteristics may have contributed to the model's performance gap in *pixel specificity* between *hold* and *hnew* slides in Experiment 2. This finding suggests that stain normalization may be a promising approach to address the challenge of dealing with a diverse dataset as in this thesis. Stain normalization is a process that standardizes the color and intensity of histopathological slides and mitigates the effects of variations. Thus, future work could explore replacing the data augmentation strategies with stain normalization methods to reduce the variability in the dataset and potentially improve model performance. Additionally, it could be evaluated on even more diverse datasets, ideally multi-institutional, to ensure that the model generalizes well to different stain preparation protocols and imaging conditions.

In addition, the evaluation of individual SwinMIL stages indicated a significant role of stage weights in balancing high- and low-level details. However, the default weights were not optimal for high-resolution segmentation tasks. Two potential directions could be investigated in the future. First, future work could experiment with making the stage weights learnable parameters rather than using predefined weights. This approach could automatically adapt the weights according to the training data, potentially leading to a better balance between high- and low-resolution stages. A second direction could be to explore changing the stage weights manually as a postprocessing step after training. One possible approach would be to visually determine how much background is included in the segmentation mask at the tissue boundaries. Accordingly, the weight of Stage 3 could be decreased to reduce oversegmentation or increased to include more context. By visually evaluating different weight combinations on a validation set, it might be possible to determine a more optimal set of weights for the specific task and data at hand.

The application of hard negative mining has successfully mitigated false positives, but there is still room for improvement. For example, it may prove beneficial to apply hard negative mining iteratively. That is, after each iteration, the model would be retrained with new negatively mined examples, further refining its ability to discriminate between cancerous and non-cancerous epithelial tissue. This approach could help to reduce false positives further and increase the robustness of the model, which is crucial with regard to its application in clinical practice.

Finally, since pretraining the Swin Transformer backbone with DINO did not yield the expected improvements, it may be beneficial to explore other architectures for pretraining, such as the Vision Transformer, and replace the backbone of SwinMIL. Given that the Vision Transformer provides a more global understanding of the image content than the Swin Transformer, it may produce better feature representations for semantic segmentation when pretrained with DINO.

8 Summary

The thesis starts by giving an overview of cancer, its diagnostic process, and the challenges associated with it. It introduces digital pathology as a new approach to cancer diagnosis and highlights the benefits of automated histological analysis through software, specifically deep learning-based computer vision algorithms. The potential of WSSS approaches is presented as a solution to the time-consuming labeling process required for supervised semantic segmentation of WSIs. Furthermore, the benefits of high-resolution segmentation for histopathological images are addressed: Deep learning models for disease prediction or classification could learn from the detailed features extracted from a high-resolution segmentation to improve their performance. In addition, a high-resolution segmentation can be used to quantify different tissue components (e.g. tumor cells, immune cells, stroma) in WSIs with higher accuracy than a low-resolution segmentation. Finally, the primary focus of the thesis was presented, which is the investigation of the WSSS model SwinMIL on high-resolution WSIs and the effect of self-supervised pretraining with DINO on a Swin Transformer backbone in the context of histopathological image segmentation.

The following Chapter 2 provides the medical and technical background for this thesis. It begins with histopathology, covering the process of sample preparation, the relevant staining techniques, and the criteria used in cancer treatment and prognosis. It also delves into the specifics of colorectal cancer and the role of digital pathology in healthcare. Afterwards, the fundamental concepts and architectures of deep neural networks are covered, including MLPs and CNNs. Furthermore, the Vision Transformer architecture, a recent development in the field, is introduced and a comparison between CNNs and Transformer-based architectures is provided. Following this, different hyperparameter optimization strategies are presented, including BOHB, which was used for the WSSS experiments in this master thesis. The final section of the chapter lays the main foundation for the related work and methodologies that follow, focusing on WSSS methods that use coarse training labels to predict the class of each pixel in an image. It discusses SSL and MIL, two methods that are particularly relevant to this task.

Chapter 3 provides a review of the existing literature and methodologies relevant to the topic of WSSS on histopathological WSIs and introduces the deep learning architectures used in this thesis. It starts with representation learning via WSI classification, which provides attention maps that can be interpreted as segmentation masks. This is followed by methods specifically designed for WSSS. However, none of these methods provide a highly detailed segmentation mask. They either perform semantic segmentation on a low resolution $> 1\,\mu\text{m/pixel}$ or suffer from oversegmentation. The chapter continues by reviewing the architectures relevant to this master's thesis research, including ResNet, Swin Transformer, and SwinMIL. Finally, it also covers the self-supervised pretraining method DINO, which is inspired by the concepts of student-teacher learning and knowledge distillation.

Chapter 4 covers several aspects of the data and presents the method used to generate pixel-wise ground truth masks as well as the experiments performed in this thesis. It starts by describing the CancerScout-CRC and NCT-CRC-HE-100K datasets used for the WSSS experiments and the pretraining experiments with DINO. Furthermore, the applied preprocessing methods are presented. This includes the registration of the IHC- and HE-stained WSIs needed to generate the pixel-wise ground truth masks. In addition, Otsu's method was explained which was used to exclude non-tissue (background) regions from all WSIs, and various types of artifacts that can appear in WSIs were discussed. Since these artifacts can affect the quality of the image and subsequent analysis, they were removed by manual annotation, except for blurred patches, which are automatically detected using the variance of the laplacian. Afterwards, the idea of generating the pixel-wise ground truth masks was described, which involved binarizing the available IHC-stained WSIs and overlaying them with their corresponding coarse regional annotations. The process involved converting the IHC stain to a grayscale image, applying a morphological opening to remove high-frequency details, applying bilateral filters to smooth the image, and finally transforming the image into a binary mask by applying simple thresholding. Finally, the chapter describes the setup of the four WSSS experiments and the four experiments for pretraining the Swin Transformer backbone with DINO. This includes the data composition as well as the intention behind each experiment.

Chapter 5 presents the main results obtained from the different experiments. The low-resolution SwinMIL model in Experiment 1 suffers significantly more from oversegmentation than the high-resolution models. However, in terms of resolution, the low-resolution segmentation masks are probably able to adequately represent most of the details present in the ground truth mask. The chapter continues by comparing the three high-resolution WSSS experiments with SwinMIL to each other using quantitative numerical metrics as well as qualitative prediction masks on the WSIs. The initial high-resolution model from Experiment 2 produced a considerable amount of false positives, especially for the epithelial tissue in *healthy* WSIs. In addition, the models with the highest *patch F1 score* did not have the highest *pixel F1 score*, i.e. the correlation between patch-level and pixel-level performance scores was not adequate. However, both problems were significantly mitigated by the application of hard negative mining in Experiment 4. The qualitative segmentation masks showed, that SwinMIL is still prone to oversegmentation in the presence of cancerous tissue. The chapter further analyzes the different stages of SwinMIL and shows that the oversegmentation problem is mainly caused by Stage 3 and partly by Stage 2. Finally, the chapter includes a comparison of the numerical results of four different DINO pretraining configurations. The pretraining of each Swin Transformer model converges regardless of the configuration. However, configurations where the dimensionality of the output vectors was decreased, converge to a lower final loss and have a higher accuracy in the subsequent k-NN classification task than the original default configuration.

Chapter 6 discusses the results of the experiments. It starts by summarizing the results and then provides possible explanations for the observations within individual experiments. The oversegmentation of the low-resolution SwinMIL model could be explained by a loss of information in the downscaling process. Nevertheless, the segmentation masks of the low-resolution model probably have sufficient resolution to capture most ground truth structures precisely. The chapter further underlines the importance of the individual SwinMIL stages in balancing high- and low-level information in the final segmentation mask. Stage 1 effectively recognizes and segments epithelial cells but cannot differentiate between cancerous and non-cancerous epithelial tissue. Stage 3 can identify whether the tissue is cancerous but at the expense of detail. The default stage weights were found to be suboptimal for the high-resolution segmentation in this thesis. The chapter continues by discussing the unexpected observation that supervised pretraining of the Swin Transformer backbone on ImageNet showed better performance than self-supervised pretraining with DINO on histopathological images. It is assumed that the minor effectiveness of DINO pretraining may be due to the employed Swin Transformer network which has distinct properties compared to the Vision Transformer used in the original paper. Finally, the effect of hard negative mining was discussed. It significantly reduced false positives in *healthy* WSIs and improved the correlation between *patch* and *pixel F1 scores*. However, the *pixel F1 score* of the best model trained with negatively mined examples was slightly lower than the reference model, although the significant improvements in *pixel specificity* compensated for this.

The final Chapter 7 provides future directions for improving the WSSS of histopathological images. It suggests the use of stain normalization to address the variance in color characteristics across different histopathology slides, which could improve the overall model performance. Furthermore, it proposes making the stage weights within the SwinMIL model learnable parameters or adapting them visually on the validation dataset to better balance high- and low-level details. The potential of iterative hard negative mining is mentioned to further reduce false positives and improve the model's ability to discriminate between cancerous and non-cancerous tissue. Finally, the possibility of exploring other architectures for DINO pretraining, such as the Vision Transformer, is suggested to improve the segmentation performance of SwinMIL.

List of Abbreviations

CAM	class activation map
CNN	Convolutional neural network
DNN	deep neural network
FFPE	formalin-fixed paraffin-embedded
HE	hematoxylin-eosin
HIPT	Hierarchical Image Pyramid Transformer
IHC	immunohistochemical
MIL	multiple-instance learning
MLP	multilayer perceptron
mpp	microns per pixel
MSI	microsatellite instability
MSS	microsatellite stable
NLP	natural language processing
PCA	principal component analysis
ReLU	rectified linear unit
ResNet	residual neural network
SGD	stochastic gradient descent
SSL	self-supervised learning
UMAP	Uniform Manifold Approximation and Projection
WSI	whole-slide image
WSSS	weakly-supervised semantic segmentation

List of Figures

List of Tables

References

[1] J. Ahn, S. Cho, and S. Kwak, "Weakly supervised learning of instance segmentation with inter-pixel relations", in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 2209–2218.

[2] R. Anil, G. Pereyra, A. Passos, R. Ormandi, G. E. Dahl, and G. E. Hinton, "Large scale distributed neural network training through online distillation", *International Conference on Learning Representations*, 2018.

[3] V. Anklin, P. Pati, G. Jaume, B. Bozorgtabar, A. Foncubierta-Rodriguez, J.-P. Thiran, M. Sibony, M. Gabrani, and O. Goksel, "Learning whole-slide segmentation from inexact and incomplete labels using tissue graphs", in *Medical Image Computing and Computer Assisted Intervention–MICCAI 2021: 24th International Conference, Strasbourg, France, September 27–October 1, 2021, Proceedings, Part II 24*, Springer, 2021, pp. 636–646.

[4] N. B. Apoorv Agnihotri. "Exploring bayesian optimization". (2020), [Online]. Available: https://distill.pub/2020/bayesian-optimization/ (visited on 03/25/2023).

[5] S. Asgari Taghanaki, K. Abhishek, J. P. Cohen, J. Cohen-Adad, and G. Hamarneh, "Deep semantic segmentation of natural and medical images: A review", *Artificial Intelligence Review*, vol. 54, pp. 137–178, 2021.

[6] R. Awan, K. Sirinukunwattana, D. Epstein, S. Jefferyes, U. Qidwai, Z. Aftab, I. Mujeeb, D. Snead, and N. Rajpoot, "Glandular morphometrics for objective grading of colorectal adenocarcinoma histology images", *Scientific reports*, vol. 7, no. 1, p. 16 852, 2017.

[7] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation", *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[8] H. Bao, L. Dong, S. Piao, and F. Wei, "Beit: Bert pre-training of image transformers", *International Conference on Learning Representations*, 2022.

[9] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization", *Advances in neural information processing systems*, vol. 24, 2011.

[10] A. G. Berman, W. R. Orchard, M. Gehrung, and F. Markowetz, "Pathml: A unified framework for whole-slide image analysis with deep learning", *medRxiv*, pp. 2021–07, 2021.

[11] J. D. Brierley, M. K. Gospodarowicz, and C. Wittekind, Eds., *TNM classification of malignant tumours*, en, 8th ed. Standards Information Network, Nov. 2016.

[12] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, "Emerging properties in self-supervised vision transformers", in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 9650–9660.

[13] L. Chan, M. S. Hosseini, and K. N. Plataniotis, "A comprehensive analysis of weakly-supervised semantic segmentation in different image domains", *International Journal of Computer Vision*, vol. 129, pp. 361–384, 2021.

[14] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs", *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

[15] R. J. Chen, C. Chen, Y. Li, T. Y. Chen, A. D. Trister, R. G. Krishnan, and F. Mahmood, "Scaling vision transformers to gigapixel images via hierarchical self-supervised learning", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 16 144–16 155.

[16] A. S. of Clinical Oncology. "Types of oncologists". (2021), [Online]. Available: https://www.cancer.net/navigating-cancer-care/cancer-basics/cancer-care-team/types-oncologists (visited on 03/23/2023).

[17] L. contributors. "Libtiff - tiff library and utilities". (2022), [Online]. Available: http://www.simplesystems.org/libtiff/ (visited on 03/25/2023).

[18] D. T. Debela, S. G. Muzazu, K. D. Heraro, M. T. Ndalama, B. W. Mesele, D. C. Haile, S. K. Kitui, and T. Manyazewal, "New approaches and procedures for cancer treatment: Current perspectives", *SAGE open medicine*, vol. 9, p. 20 503 121 211 034 366, 2021.

[19] N. Dimitriou, O. Arandjelović, and P. D. Caie, "Deep learning for whole slide image analysis: An overview", *Frontiers in medicine*, vol. 6, p. 264, 2019.

[20] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale", *International Conference on Learning Representations*, 2021.

[21] S. Falkner, A. Klein, and F. Hutter, "Bohb: Robust and efficient hyperparameter optimization at scale", in *International Conference on Machine Learning*, PMLR, 2018, pp. 1437–1446.

[22] N. Farahani, A. V. Parwani, L. Pantanowitz, *et al.*, "Whole slide imaging in pathology: Advantages, limitations, and emerging perspectives", *Pathol Lab Med Int*, vol. 7, no. 23-33, p. 4321, 2015.

[23] R. Garnett, *Bayesian Optimization*. Cambridge University Press, 2023.

[24] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks", in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

[25] U. Göttingen. "Forschung". (2022), [Online]. Available: https://pathologie.umg.eu/forschung/ (visited on 03/25/2023).

[26] U. Göttingen. "Künstliche intelligenz zur vorhersage von mutationen in tumoren". (2022), [Online]. Available: https://pathologie.umg.eu/forschung/kuenstliche-intelligenz-zur-vorhersage-von-mutationen-in-tumoren/ (visited on 03/25/2023).

[27] W. Gu, S. Wang, S. Zhao, L. Wan, and Z. Zhu, "Histosegrest: A weakly supervised learning method for histopathology image segmentation", in *2022 the 5th International Conference on Image and Graphics Processing (ICIGP)*, 2022, pp. 189–195.

[28] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification", in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[30] G. Hinton, J. Dean, and O. Vinyals, "Distilling the knowledge in a neural network", Mar. 2014, pp. 1–9.

[31] D. J. Ho, D. V. Yarlagadda, T. M. D'Alfonso, M. G. Hanna, A. Grabenstetter, P. Ntiamoah, E. Brogi, L. K. Tan, and T. J. Fuchs, "Deep multi-magnification networks for multi-class breast cancer image segmentation", *Computerized Medical Imaging and Graphics*, vol. 88, p. 101 866, 2021.

[32] L. Hou, D. Samaras, T. M. Kurc, Y. Gao, J. E. Davis, and J. H. Saltz, "Patch-based convolutional neural network for whole slide tissue image classification", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2424–2433.

[33] Z. Huang, X. Wang, J. Wang, W. Liu, and J. Wang, "Weakly-supervised semantic segmentation network with deep seeded region growing", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7014–7023.

[34] D. B. Hulsken. "Philips isyntax for digital pathology". (2016), [Online]. Available: https://thepathologist.com/fileadmin/issues/App_Notes/0016-024-app-note-Philips_iSyntax_for_Digital_Pathology.pdf (visited on 03/25/2023).

[35] D. Imaging and C. in Medicine. "Dicom whole slide imaging (wsi)". (2022), [Online]. Available: https://dicom.nema.org/dicom/dicomwsi (visited on 03/23/2023).

[36] N. C. Institute. "Understanding cancer prognosis". (2019), [Online]. Available: https://www.cancer.gov/about-cancer/diagnosis-staging/prognosis (visited on 03/23/2023).

[37] N. C. Institute. "Colon, rectosigmoid, and rectum equivalent terms and definitions". (2020), [Online]. Available: https://seer.cancer.gov/tools/solidtumor/Colon_STM.pdf (visited on 03/23/2023).

[38] N. C. Institute. "Cancer staging". (2022), [Online]. Available: https://www.cancer.gov/about-cancer/diagnosis-staging/staging (visited on 03/23/2023).

[39] N. C. Institute. "Tumor grade". (2022), [Online]. Available: https://www.cancer.gov/about-cancer/diagnosis-staging/diagnosis/tumor-grade (visited on 03/23/2023).

[40] N. C. Institute. "What is cancer?" (2023), [Online]. Available: https://www.cancer.gov/about-cancer/understanding/what-is-cancer (visited on 03/23/2023).

[41] Z. Jia, X. Huang, I. Eric, C. Chang, and Y. Xu, "Constrained deep weak supervision for histopathology image segmentation", *IEEE transactions on medical imaging*, vol. 36, no. 11, pp. 2376–2388, 2017.

[42] J. S. Juul, N. Hornung, B. Andersen, S. Laurberg, F. Olesen, and P. Vedsted, "The value of using the faecal immunochemical test in general practice on patients presenting with non-alarm symptoms of colorectal cancer", *British journal of cancer*, vol. 119, no. 4, pp. 471–479, 2018.

[43] Z. Karnin, T. Koren, and O. Somekh, "Almost optimal exploration in multi-armed bandits", in *International Conference on Machine Learning*, PMLR, 2013, pp. 1238–1246.

[44] J. N. Kather, N. Halama, and A. Marx, *100,000 histological images of human colorectal cancer and healthy tissue*, 2018. DOI: 10.5281/zenodo.1214456.

[45] M. Khened, A. Kori, H. Rajkumar, G. Krishnamurthi, and B. Srinivasan, "A generalized deep learning framework for whole-slide image segmentation and analysis", *Scientific reports*, vol. 11, no. 1, pp. 1–14, 2021.

[46] N. A. Koohbanani, "Working with scarce annotations in computational pathology", Ph.D. dissertation, University of Warwick, 2020.

[47] D. Krebsforschungszentrum. "Mikrosatelliteninstabilität: Was ist das eigentlich?" (2022), [Online]. Available: https://www.krebsinformationsdienst.de/fachkreise/nachrichten/2022/fk09-mikrosatelliteninstabilitaet-bei-krebs.php (visited on 03/25/2023).

[48] V. Kumar, A. K. Abbas, N. Fausto, and J. C. Aster, *Robbins and Cotran pathologic basis of disease, professional edition e-book*. Elsevier health sciences, 2014.

[49] K. Larson, H. H. Ho, P. L. Anumolu, and T. M. Chen, "Hematoxylin and eosin tissue stain in mohs micrographic surgery: A review", *Dermatologic surgery*, vol. 37, no. 8, pp. 1089–1099, 2011.

[50] M. Lerousseau, M. Vakalopoulou, M. Classe, J. Adam, E. Battistella, A. Carré, T. Estienne, T. Henry, E. Deutsch, and N. Paragios, "Weakly supervised multiple instance learning histopathological tumor segmentation", in *Medical Image Computing and Computer Assisted Intervention–MICCAI 2020: 23rd International Conference, Lima, Peru, October 4–8, 2020, Proceedings, Part V 23*, Springer, 2020, pp. 470–479.

[51] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization", *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6765–6816, 2017.

[52] G. Litjens, P. Bandi, B. Ehteshami Bejnordi, O. Geessink, M. Balkenhol, P. Bult, A. Halilovic, M. Hermsen, R. van de Loo, R. Vogels, *et al.*, "1399 h&e-stained sentinel lymph node sections of breast cancer patients: The camelyon dataset", *GigaScience*, vol. 7, no. 6, giy065, 2018.

[53] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. "Swin transformer". (2021), [Online]. Available: https://github.com/microsoft/Swin-Transformer (visited on 03/25/2023).

[54] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows", in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 10 012–10 022.

[55] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[56] M. Y. Lu, D. F. Williamson, T. Y. Chen, R. J. Chen, M. Barbieri, and F. Mahmood, "Data-efficient and weakly supervised computational pathology on whole-slide images", *Nature biomedical engineering*, vol. 5, no. 6, pp. 555–570, 2021.

[57] S. Mac Giobuin, D. Kavanagh, E. Myers, A. Doherty, C. Quinn, T. Crotty, D. Evoy, and E. McDermott, "The significance of immunohistochemistry positivity in sentinel nodes which are negative on haematoxylin and eosin in breast cancer", *European Journal of Surgical Oncology (EJSO)*, vol. 35, no. 12, pp. 1257–1260, 2009.

[58] M. Macenko, M. Niethammer, J. S. Marron, D. Borland, J. T. Woosley, X. Guan, C. Schmitt, and N. E. Thomas, "A method for normalizing histology slides for quantitative analysis", in *2009 IEEE international symposium on biomedical imaging: from nano to macro*, IEEE, 2009, pp. 1107–1110.

[59] A. Maćkiewicz and W. Ratajczak, "Principal components analysis (pca)", *Computers & Geosciences*, vol. 19, no. 3, pp. 303–342, 1993.

[60] L. McInnes, J. Healy, N. Saul, and L. Großberger, "Umap: Uniform manifold approximation and projection", *Journal of Open Source Software*, vol. 3, no. 29, p. 861, 2018. DOI: 10.21105/joss.00861. [Online]. Available: https://doi.org/10.21105/joss.00861.

[61] L. E. Morrison, M. R. Lefever, H. N. Lewis, M. J. Kapadia, and D. R. Bauer, "Conventional histological and cytological staining with simultaneous immunohistochemistry enabled by invisible chromogens", *Laboratory Investigation*, vol. 102, no. 5, pp. 545–553, 2022.

[62] G. O'Dowd, S. Bell, and S. Wright, *Wheater's Pathology: A Text, Atlas and Review of Histopathology E-Book*. Elsevier Health Sciences, 2019.

[63] K. Okada, "Points to notice during the diagnosis of soft tissue tumors according to the "clinical practice guideline on the diagnosis and treatment of soft tissue tumors"", *Journal of Orthopaedic Science*, vol. 21, no. 6, pp. 705–712, 2016.

[64] W. H. Organization. "Cancer". (2023), [Online]. Available: https://www.who.int/en/news-room/fact-sheets/detail/cancer (visited on 03/23/2023).

[65] N. Otsu, "A threshold selection method from gray-level histograms", *IEEE transactions on systems, man, and cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[66] Z. Qian, K. Li, M. Lai, E. I.-C. Chang, B. Wei, Y. Fan, and Y. Xu, "Transformer based multiple instance learning for weakly supervised histopathology image segmentation", in *Medical Image Computing and Computer Assisted Intervention–MICCAI 2022: 25th International Conference, Singapore, September 18–22, 2022, Proceedings, Part II*, Springer, 2022, pp. 160–170.

[67] J. Ramos-Vara and M. Miller, "When tissue antigens and antibodies get along: Revisiting the technical aspects of immunohistochemistry—the red, brown, and blue technique", *Veterinary pathology*, vol. 51, no. 1, pp. 42–87, 2014.

[68] J. A. Ramos-Vara, "Technical aspects of immunohistochemistry", *Veterinary pathology*, vol. 42, no. 4, pp. 405–426, 2005.

[69] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation", in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, Springer, 2015, pp. 234–241.

[70] M. H. Ross and W. Pawlina, *Histology - A Text and Atlas: with Correlated Cell and Molecular Biology*. Wolters Kluwer/Lippincott Williams & Wilkins Health, 2011, ISBN: 978-0-781-77200-6.

[71] A. C. Society. "Colorectal cancer signs and symptoms". (2020), [Online]. Available: https://www.cancer.org/cancer/colon-rectal-cancer/detection-diagnosis-staging/signs-and-symptoms.html (visited on 03/23/2023).

[72] A. C. Society. "Cancer staging". (2022), [Online]. Available: https://www.cancer.org/treatment/understanding-your-diagnosis/staging.html (visited on 03/23/2023).

[73] R. Strudel, R. Garcia, I. Laptev, and C. Schmid, "Segmenter: Transformer for semantic segmentation", in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 7262–7272.

[74] K. S. Suvarna, C. Layton, and J. D. Bancroft, *Bancroft's Theory and Practice of Histological Techniques*. Edinburgh, New York: Elsevier Health Sciences, 2018, ISBN: 978-0-702-06886-7.

[75] C. R. UK. "Diagram showing t stages of bowel cancer". (2015), [Online]. Available: https://commons.wikimedia.org/wiki/File:Diagram_showing_T_stages_of_bowel_cancer_CRUK_276.svg (visited on 03/25/2023).

[76] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need", *Advances in neural information processing systems*, vol. 30, 2017.

[77] M. R. Wick, "The hematoxylin and eosin stain in anatomic pathology—an often-neglected focus of quality assurance in the laboratory", in *Seminars in diagnostic pathology*, Elsevier, vol. 36, 2019, pp. 303–311.

[78] C. P. Wild, B. W. Stewart, and C. Wild, *World cancer report 2014*. World Health Organization Geneva, Switzerland, 2014.

[79] D. Wittekind, "Traditional staining for routine diagnostic pathology including the role of tannic acid. 1. value and limitations of the hematoxylin-eosin stain", *Biotechnic & histochemistry*, vol. 78, no. 5, pp. 261–270, 2003.

[80] G. Xu, Z. Song, Z. Sun, C. Ku, Z. Yang, C. Liu, S. Wang, J. Ma, and W. Xu, "Camel: A weakly supervised learning framework for histopathology image segmentation", in *Proceedings of the IEEE/CVF International Conference on computer vision*, 2019, pp. 10 682–10 691.

[81] Y. Xu, J.-Y. Zhu, I. Eric, C. Chang, M. Lai, and Z. Tu, "Weakly supervised histopathology cancer image segmentation and classification", *Medical image analysis*, vol. 18, no. 3, pp. 591–604, 2014.

[82] M. Zhang, Y. Zhou, J. Zhao, Y. Man, B. Liu, and R. Yao, "A survey of semi-and weakly supervised semantic segmentation of images", *Artificial Intelligence Review*, vol. 53, pp. 4259–4288, 2020.

[83] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2921–2929.

Appendices

A DNN Architectures

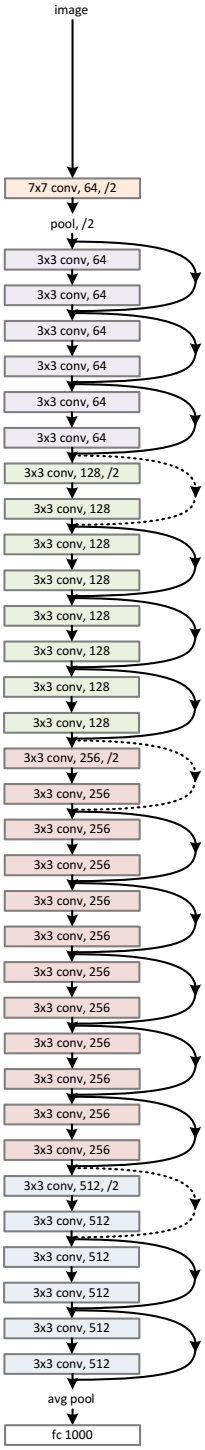
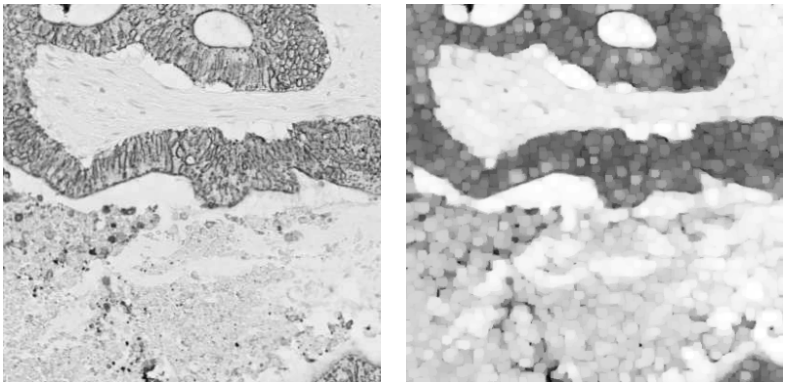


Figure A.1: Illustration of the ResNet-34 architecture. It has 34 layers in total and connects a series of 16 residual blocks employing 3×3 convolutions. At the end of the network, a global average pooling layer is applied to each feature map to create a one-dimensional feature vector.

B Generation of Binary IHC Masks



(a) Grayscale IHC Patch (b) Opening Effect

Figure B.1: Effect of morphological opening on a grayscale IHC patch containing epithelial tissue. The regions with epithelial tissue and the background become more homogeneous and contain less high-frequency information.

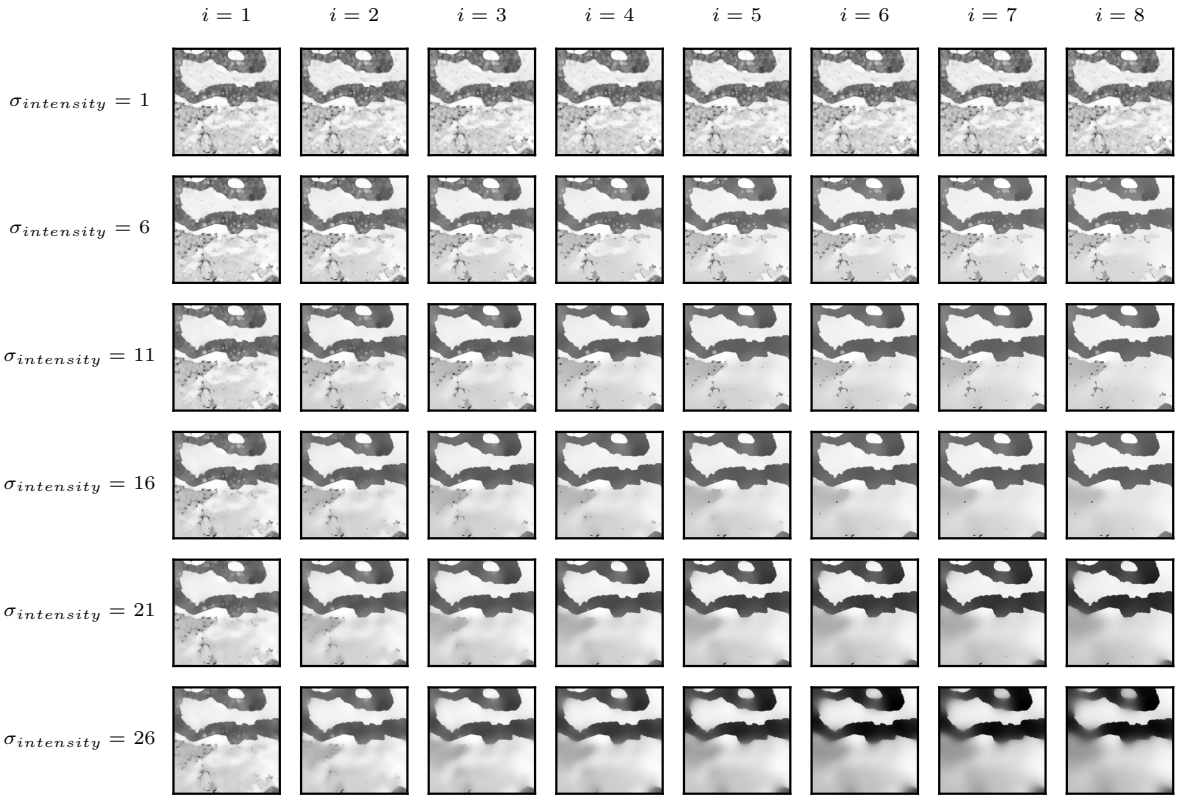


Figure B.2: Iterative bilateral filtering with different values for $\sigma_{intensity}$ applied on a grayscale IHC patch containing epithelial tissue. Each row represents a new trial with a different value for $\sigma_{intensity}$. The parameter σ_{space} was set to 15 for all trials. Iteration i represents the number of bilateral filters applied to the patch.

(a) τ =150 (b) τ =160 (c) τ =170 (d) τ =180 (e) τ =190

Figure B.3: Binary masks obtained by different intensity thresholds τ on one example patch. The thresholding was applied on the grayscale IHC patches after iterative bilateral filtering. The binary masks for this patch are identical for all thresholds between 160 and 180

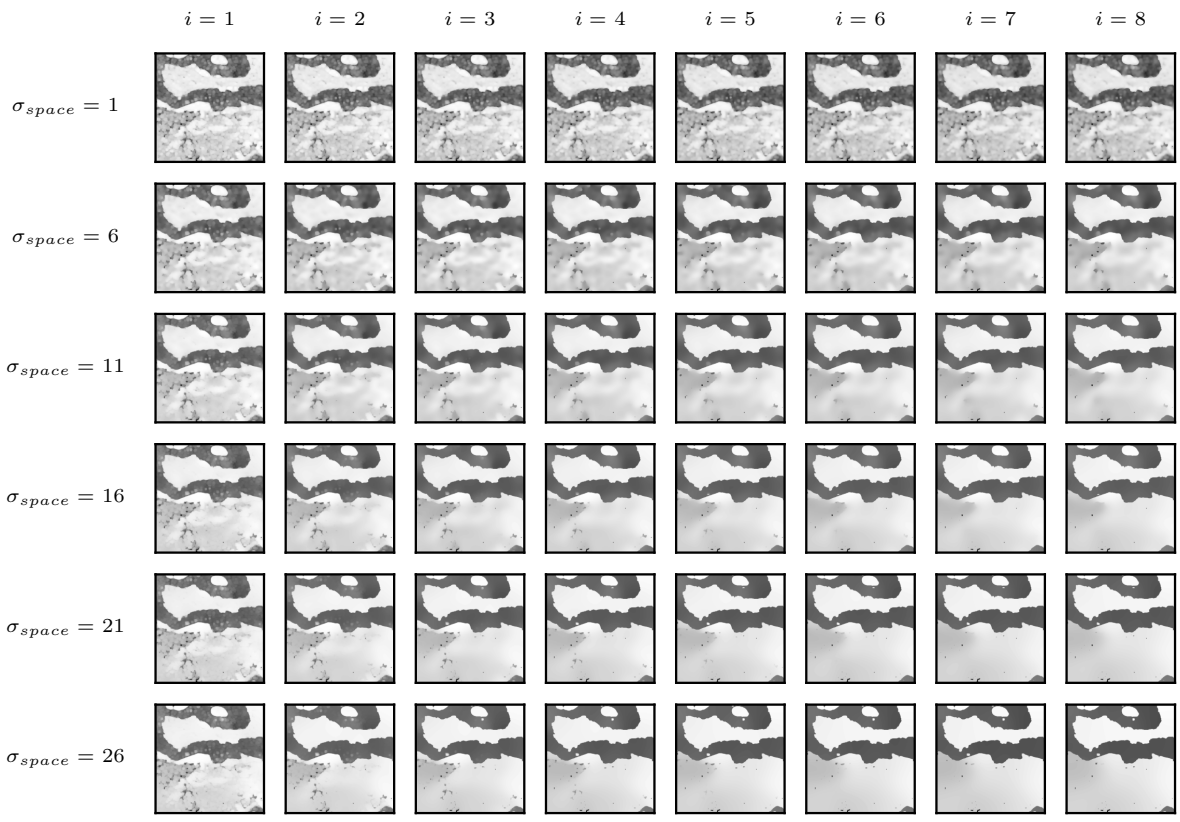
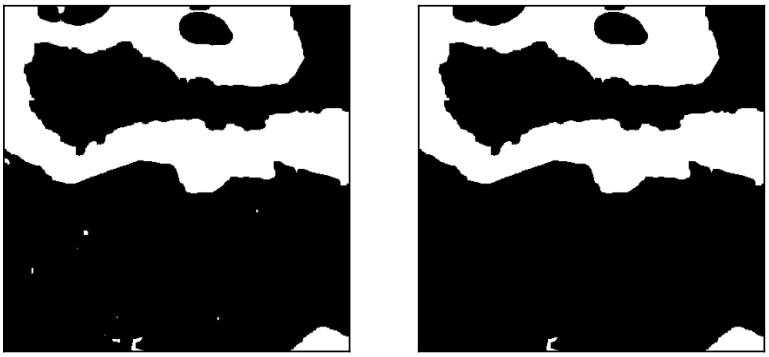


Figure B.4: Iterative bilateral filtering with different values for σ_{space} applied on a grayscale IHC patch containing epithelial tissue. Each row represents a new trial with a different value for σ_{space}. The parameter $\sigma_{intensity}$ was set to 15 for all trials. Iteration i represents the number of bilateral filters applied to the patch.



(a) Binary IHC Patch (b) Removed Noise

Figure B.5: Noise removal by applying a connected-component analysis on one example patch. After iterative bilateral filtering and thresholding, the binary mask in (a) still contains small artifacts that do not represent epithelial tissue. These were removed via connected-component analysis in (b).

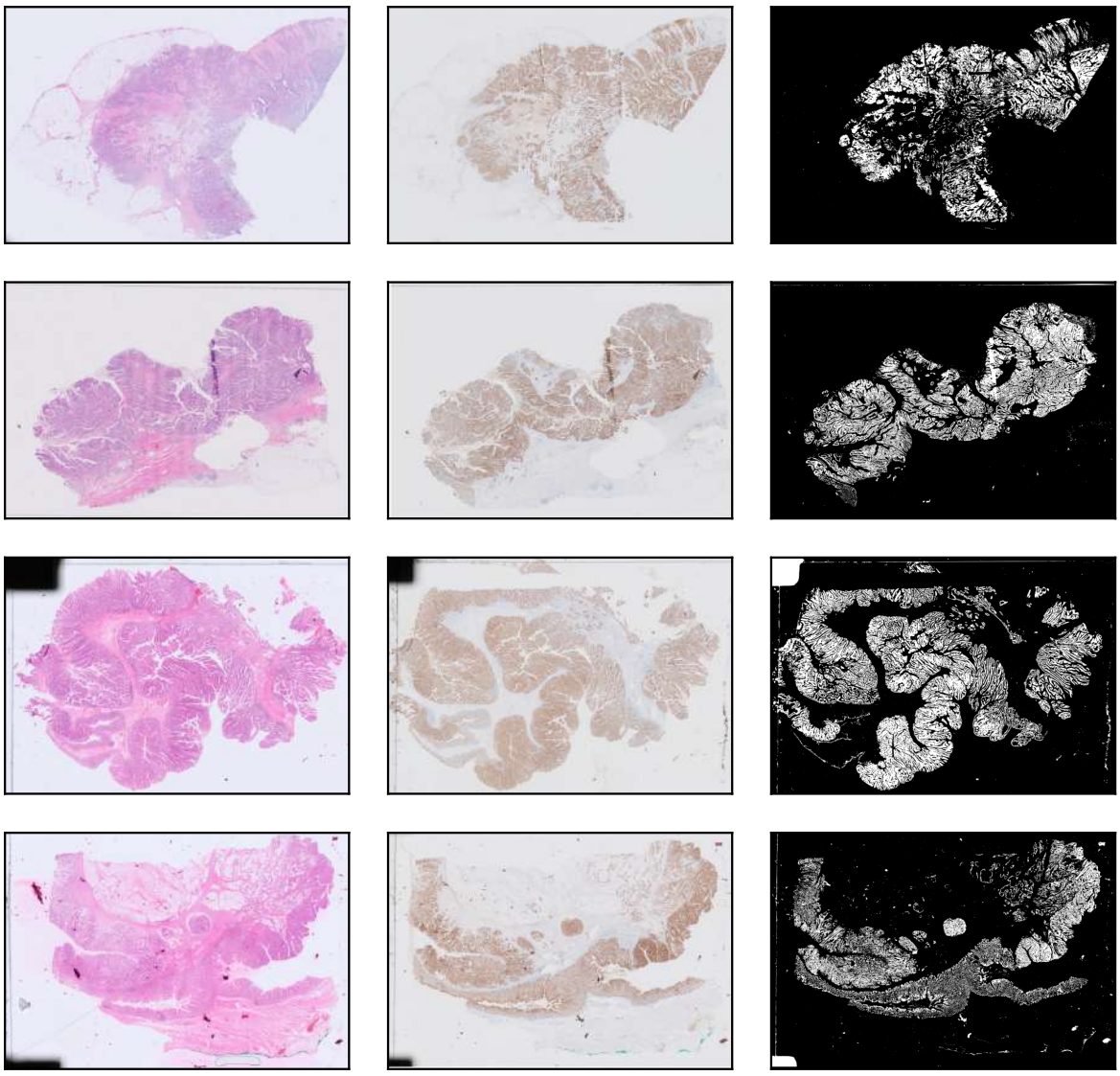
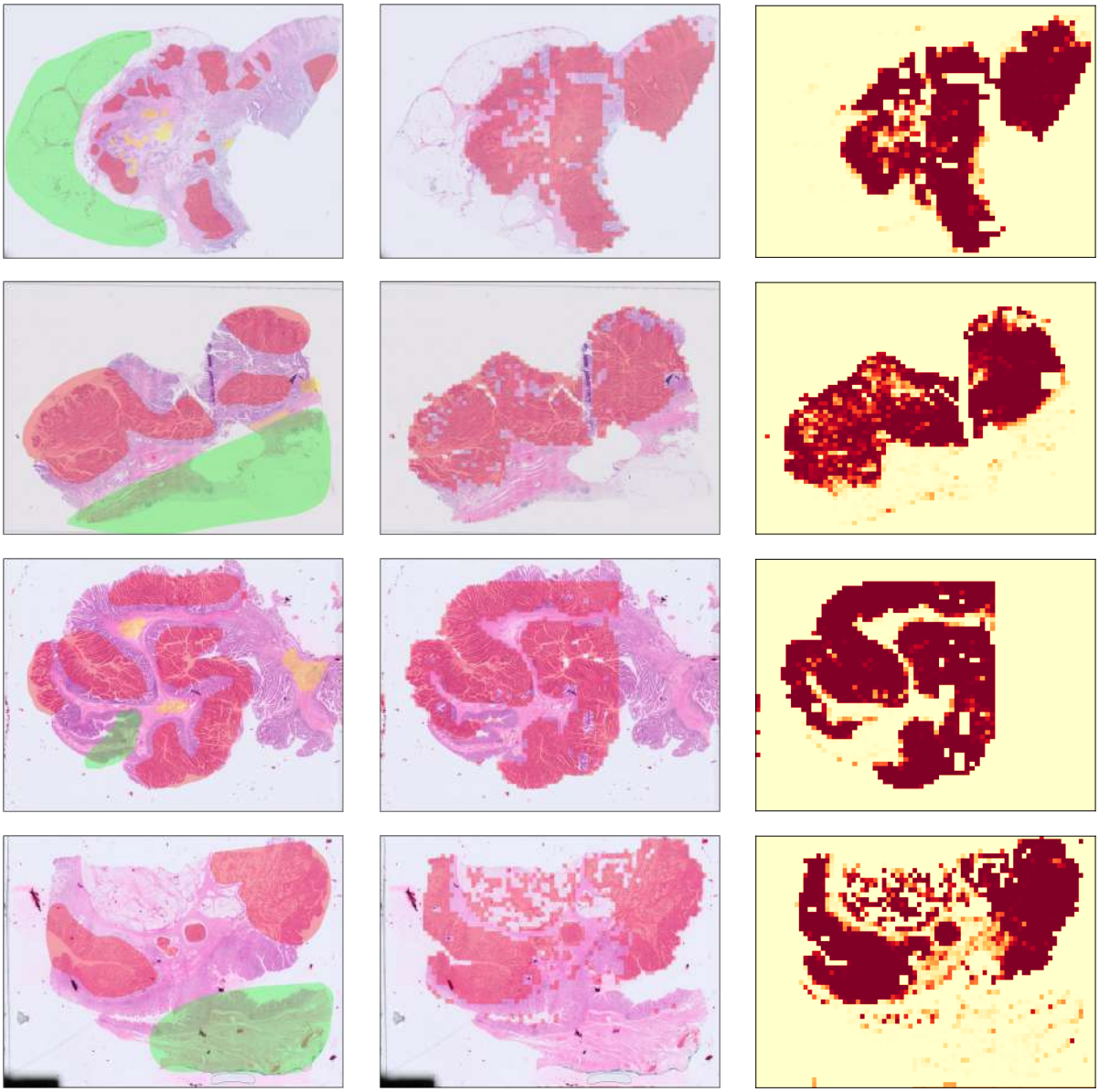


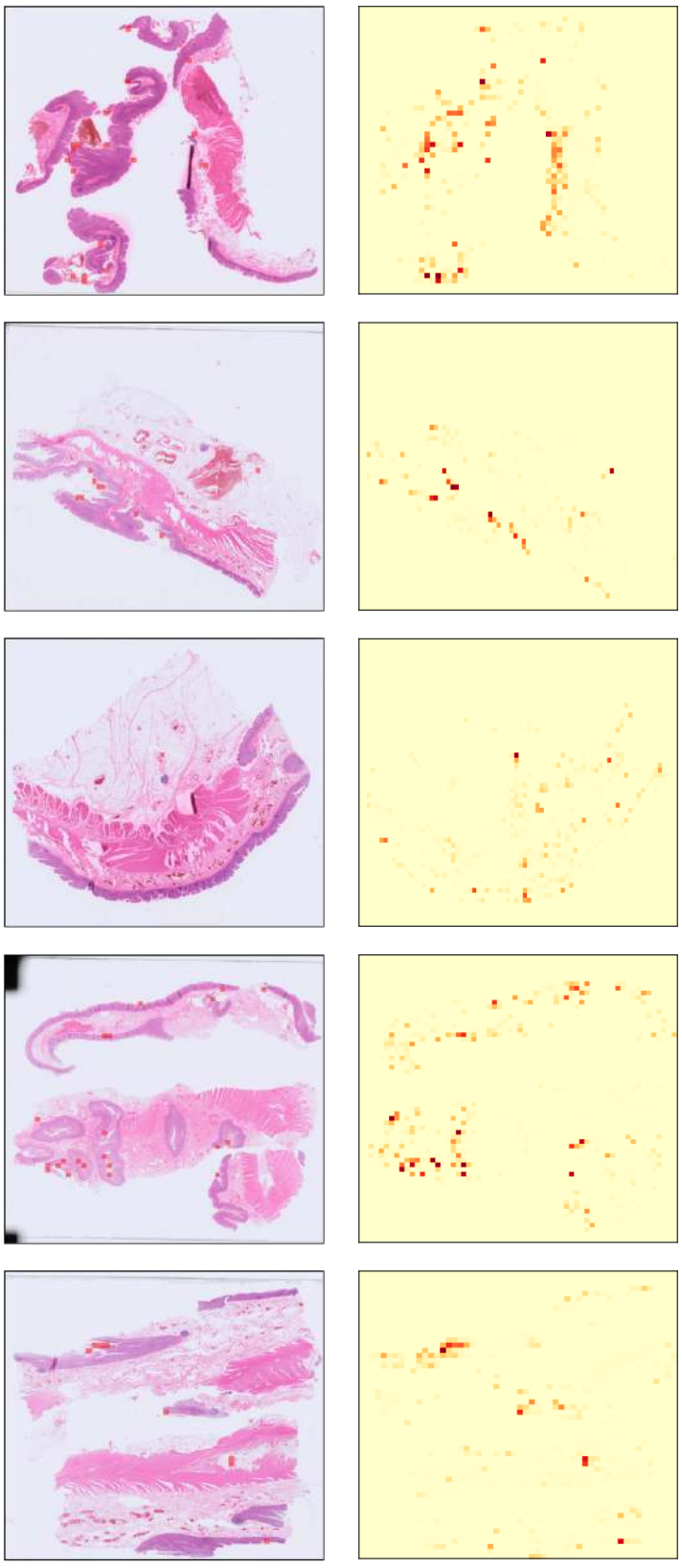
Figure B.6: Generated ground truth mask for the epithelial tissue in all four *cancerous* WSIs. The HE stained slides are shown in (a), the corresponding IHC stains in (b), and the generated binary mask in (c).

C Qualitative Results of Baseline



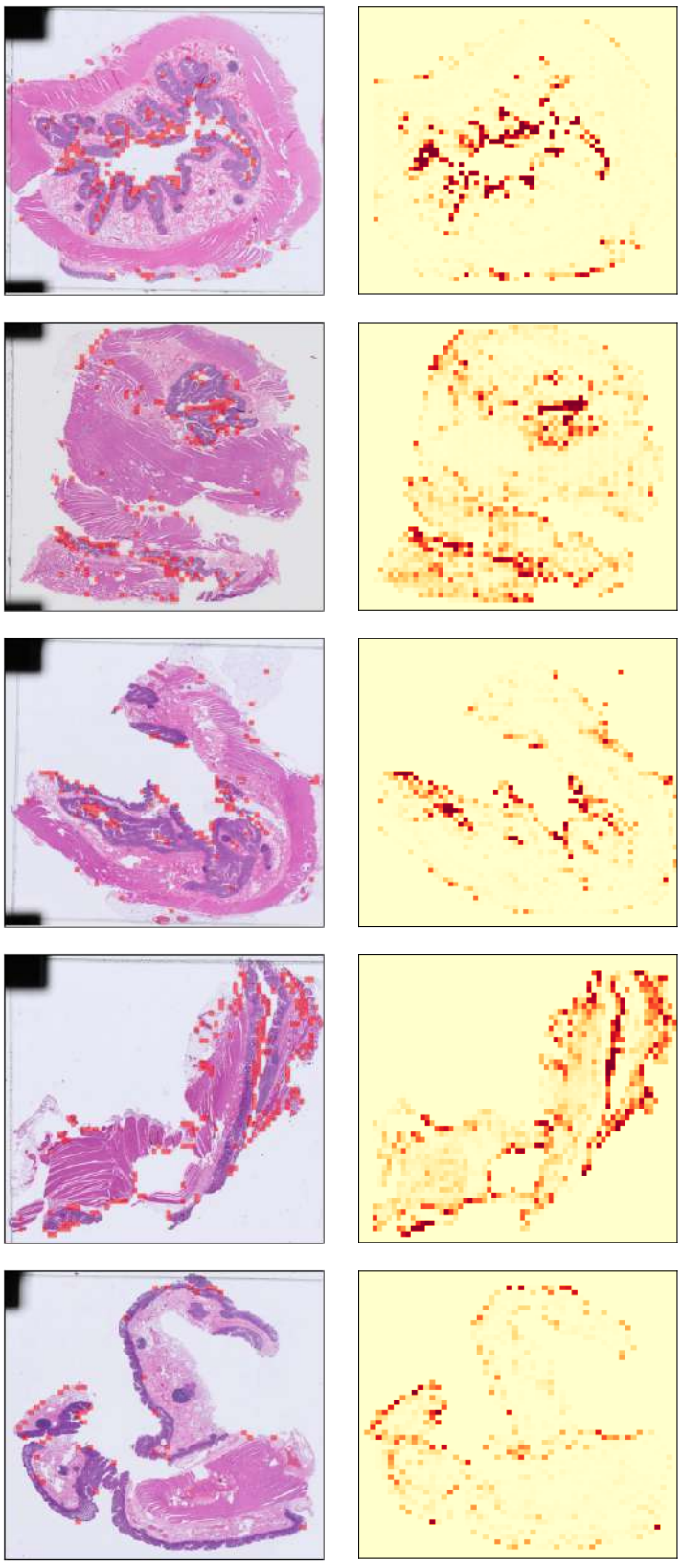
(a) Coarse Ground Truth (b) Prediction (Binary) (c) Prediction (Heatmap)

Figure C.1: Qualitative prediction masks of the best baseline model on all *cancerous* WSIs from the test set. The coarse annotation masks are shown in (a). The binary prediction of the model is superimposed on the WSIs in (b) and a raw heatmap of the prediction values is presented in (c).



(a) Prediction (Binary) (b) Prediction (Heatmap)

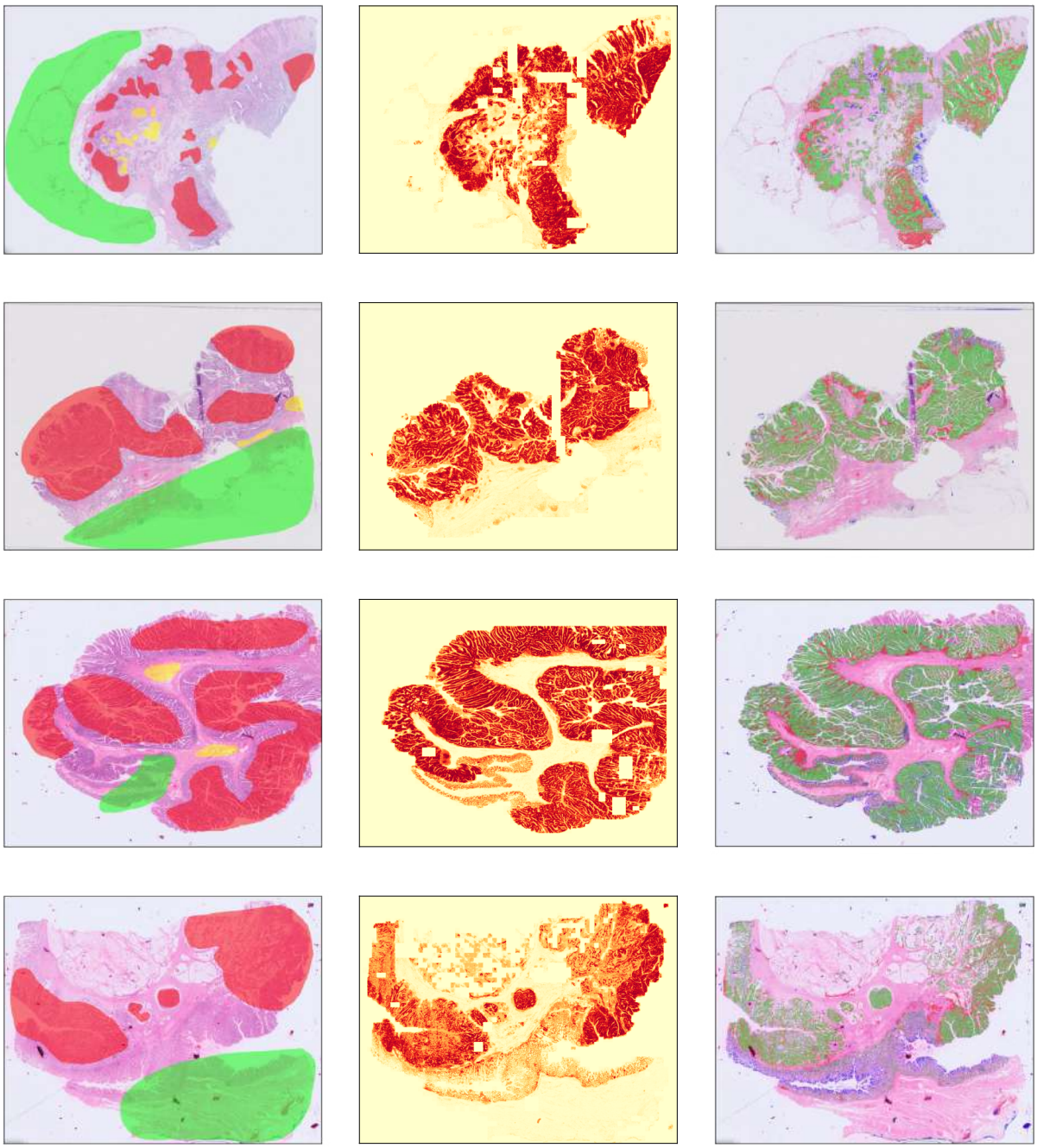
Figure C.2: Qualitative prediction mask of the best baseline model on the *hnew* WSIs from the test set. The binary prediction of the model is superimposed on the WSIs in (a) and a raw heatmap of the prediction values is presented in (b).



(a) Prediction (Binary) (b) Prediction (Heatmap)

Figure C.3: Qualitative prediction mask of the best baseline model on the *hold* WSIs from the test set. The binary prediction of the model is superimposed on the WSIs in (a) and a raw heatmap of the prediction values is presented in (b).

D Qualitative Results of SwinMIL Experiments

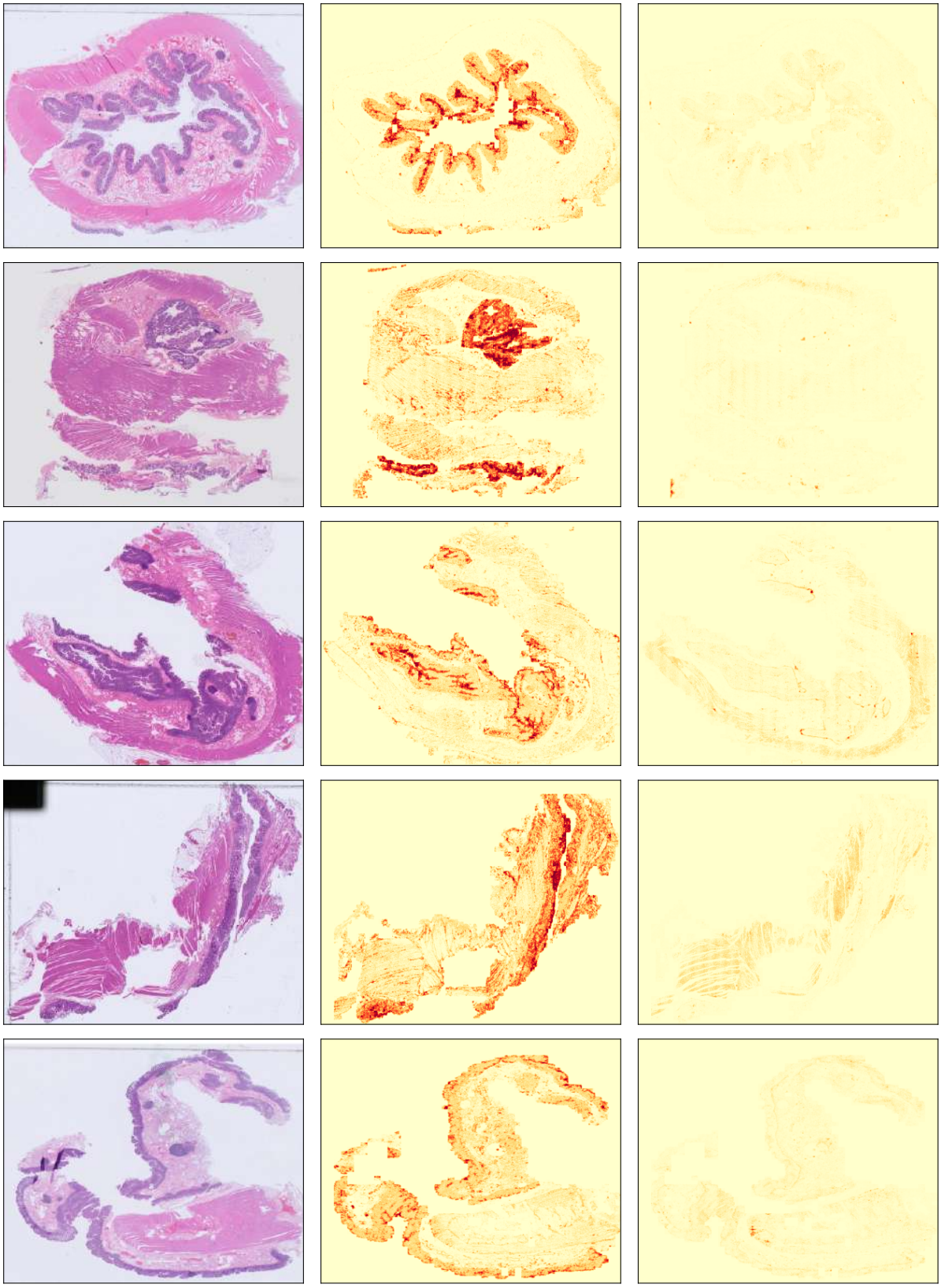


(a) Coarse Ground Truth (b) Prediction (Heatmap) (c) Prediction + Pixelwise GT

Figure D.1: Prediction masks of the best reference model on 400×400 patches from Experiment 2. The coarse annotation masks are shown in (a) and a heatmap with the raw prediction values is shown in (b). An overlay of the prediction masks with the corresponding ground truth masks is presented in (c). Regions where the model correctly detects cancerous tissue (true positives) are colored green. False positive predictions are colored red and false negatives are colored blue.

(a) Healthy WSI (b) Prediction Reference (c) Prediction Negative Mining

Figure D.2: Comparison of predictions between the high-resolution reference model from Experiment 2 and the model trained with hard negative examples in Experiment 4. Predictions are shown for all *hnew* WSIs from the test set and were obtained using the best models from both experiments.



(a) Healthy WSI (b) Prediction Reference (c) Prediction Negative Mining

Figure D.3: Comparison of predictions between the high-resolution reference model from Experiment 2 and the model trained with hard negative examples in Experiment 4. Predictions are shown for all *hold* WSIs from the test set and were obtained using the best models from both experiments.

E Analysis of SwinMIL Stages

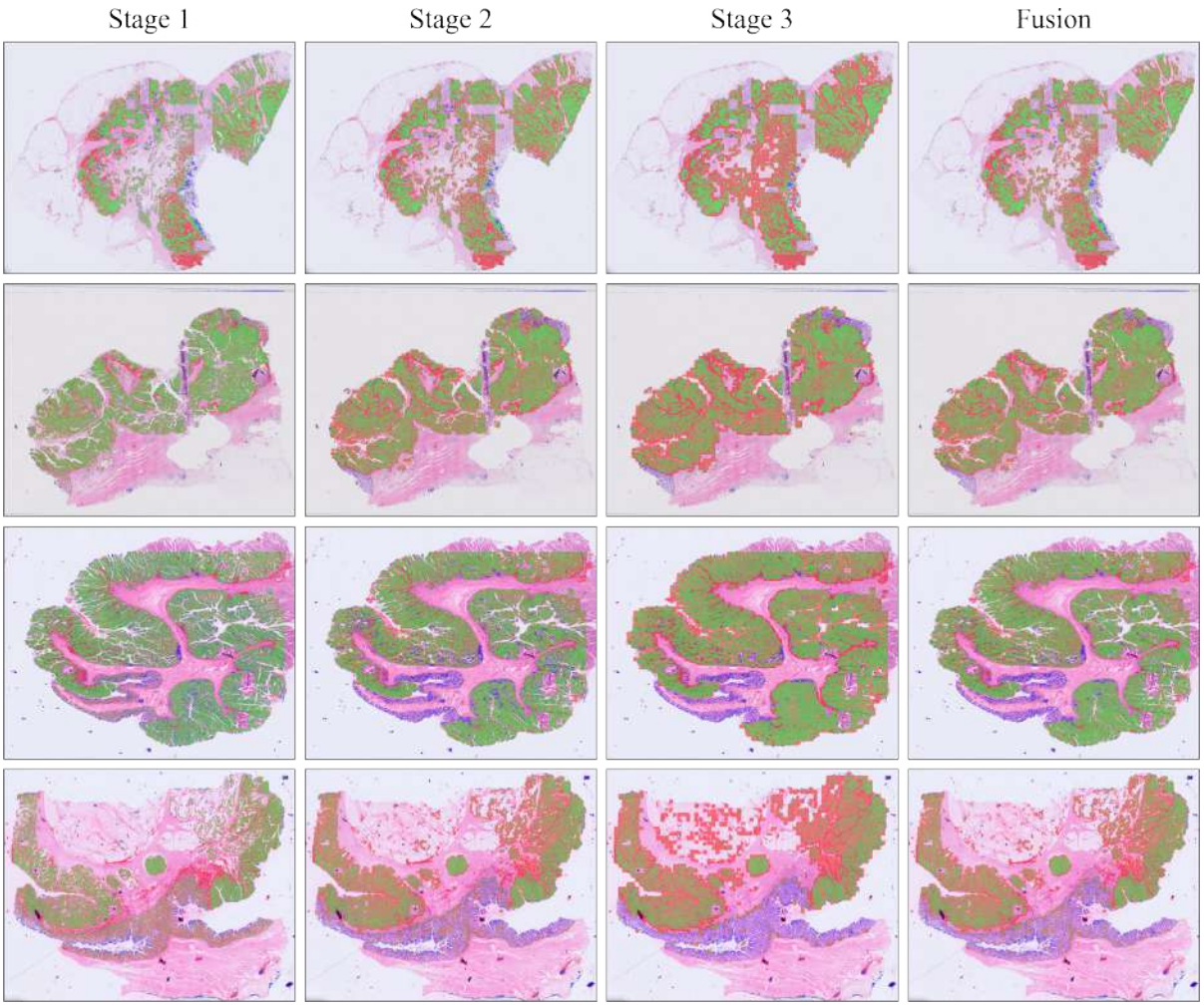


Figure E.1: Comparison of the segmentation masks across all SwinMIL stages on the *cancerous* WSIs from the test set. The prediction masks were obtained from the reference model on patch size 400×400 from Experiment 2. Each subfigure shows an overlay of the stage predictions with the corresponding ground truth mask. True positive predictions are colored green, false positives red, and false negatives blue.

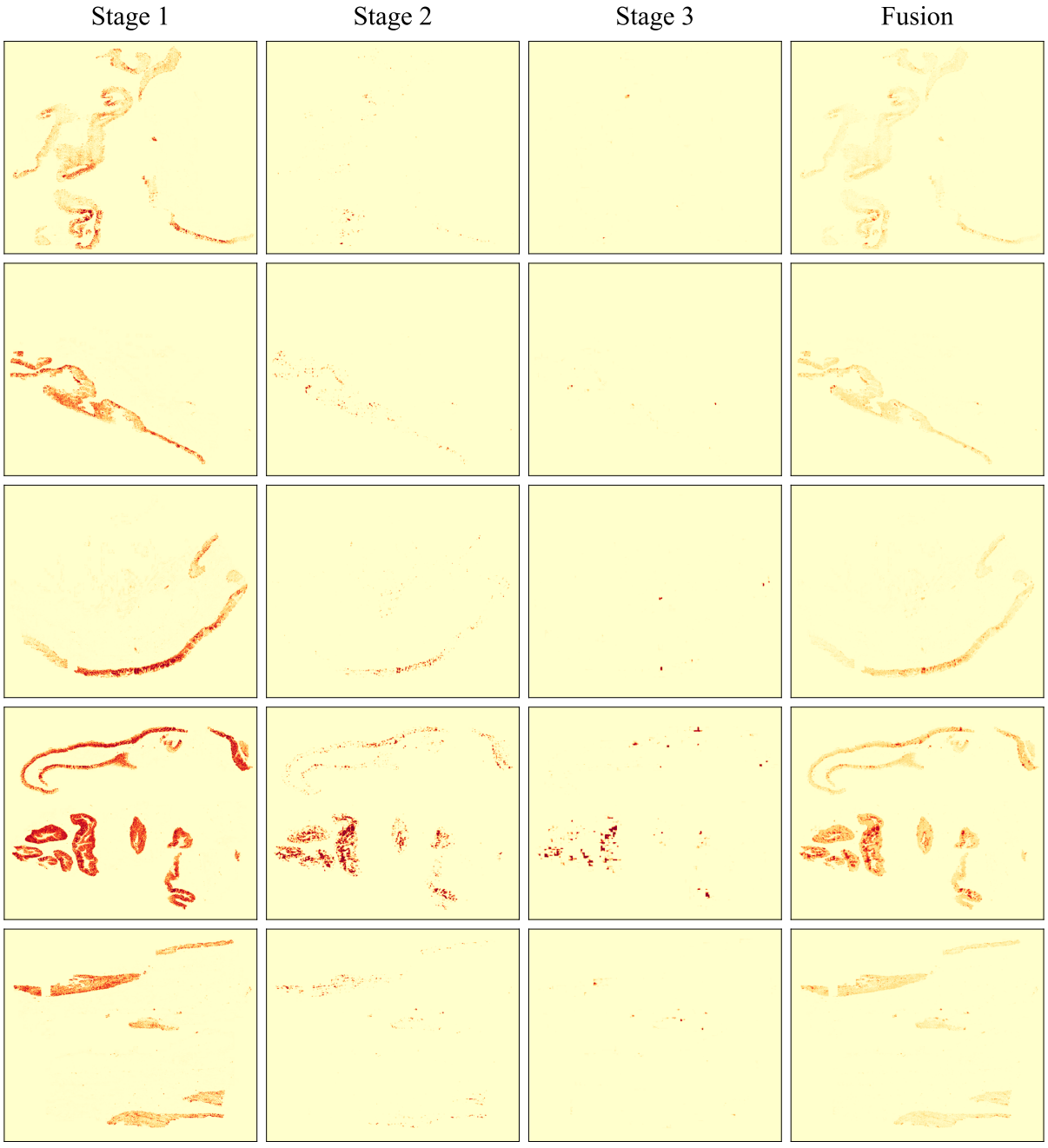


Figure E.2: Comparison of the segmentation masks across all SwinMIL stages on the *hnew* WSIs from the test set. The prediction masks were obtained from the reference model on patch size 400×400 from Experiment 2. Each row represents one *hnew* WSI. The heatmap with the raw prediction values is visualized for each stage.

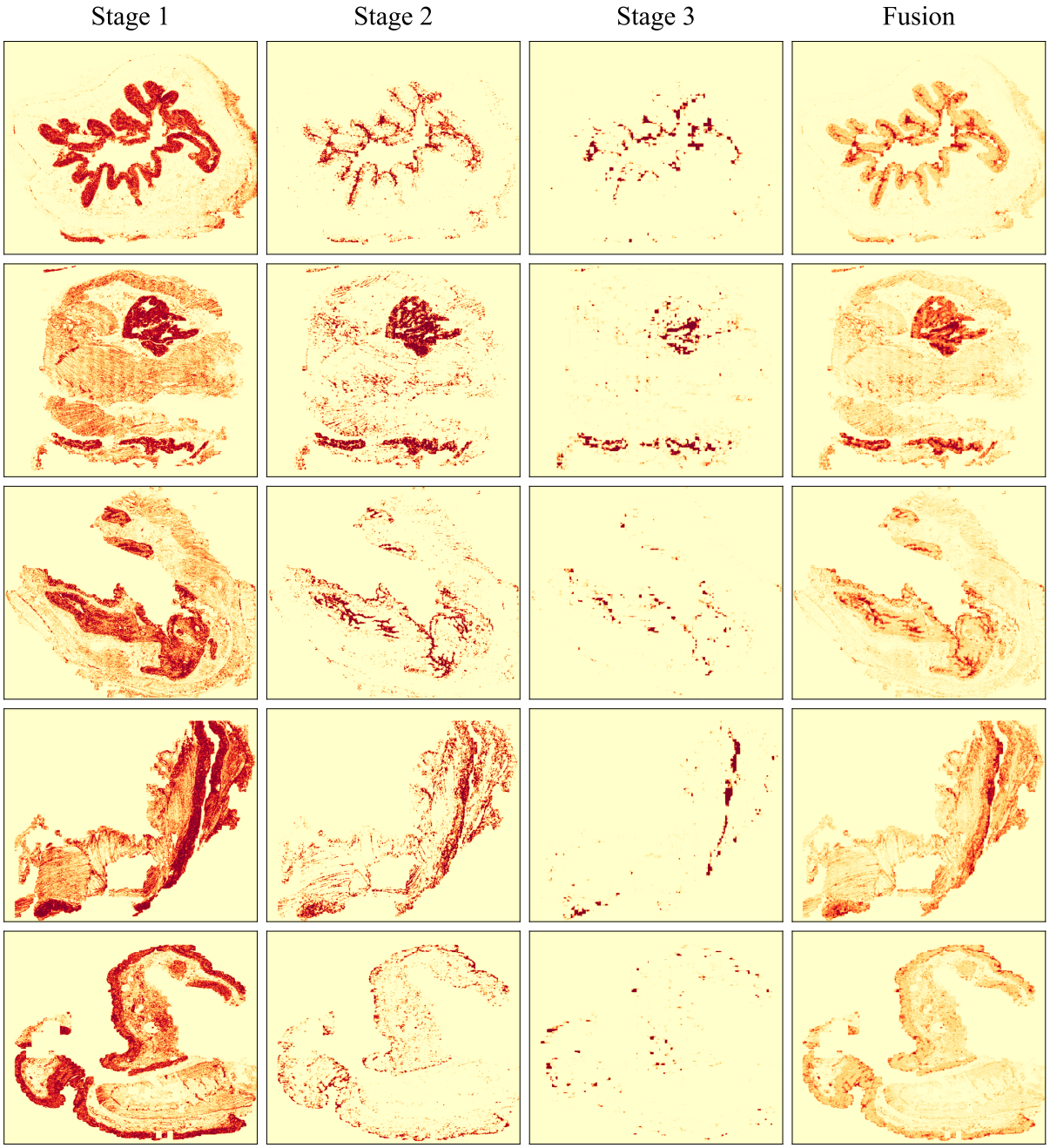


Figure E.3: Comparison of the segmentation masks across all SwinMIL stages on the *hold* WSIs from the test set. The prediction masks were obtained from the reference model on patch size 400×400 from Experiment 2. Each row represents one *hold* WSI. The heatmap with the raw prediction values is visualized for each stage.