POLITECNICO DI TORINO

Master's Degree in Computer Engineering

Master's Degree Thesis

Self-Supervision For RCC Subtyping

Supervisors

Prof. Di Cataldo SANTA

Dr. Ponzio FRANCESCO

Prof. Descombes XAVIER

Candidate

Mohamad MOHAMAD

02 2023

Summary

Cancer is a disease that occurs when abnormal cells grow and spread uncontrollably in the body. There are various types of cancer, each with their own unique characteristics and treatment options. Renal cell carcinoma (RCC) is a form of kidney cancer that can be classified into several histological subtypes. RCC is the most common type of kidney cancer in adults, making up roughly 90% of cases. Accurately identifying the tumor subtype is critical as treatment approaches and prognosis may differ based on the subtype and stage of the disease. Some of the major subtypes of RCC include clear cell carcinoma (ccRCC), papillary (pRCC), chromophobe, oncocytoma, and others. Deep learning models, specifically convolutional neural networks (CNNs), have shown great promise in the medical domain for classifying, segmenting, and detecting objects in images. However, the medical field poses unique challenges due to the lack of large and diverse datasets like ImageNet. Self-supervised techniques have been developed to address this issue, but most proposed methods for histopathological images do not fully exploit the domain data property. While they utilize the different magnifications present in whole slide images, they miss the fact that those various fields of view are interconnected. To address this gap, we propose a new self-supervised task that aims to interconnect different magnification levels of histopathological images. Our proposed task requires the model to localize a tile inside a global patch, the image representing the tile at a higher resolution is extracted from a magnification level higher than that of the global patch. Both the higher-resolution image and the global patch are given to the network to perform the localization pre-text task. We present different possible formulations for this problem and compare our method to a newly introduced state-of-the-art pretext task. Our results show that our method performs on par or better than its counterpart and outperforms models pre-trained on ImageNet. We also discuss the challenges faced during training and how we handle creating and processing our dataset. Finally, for our future work, we highlight the potential of our proposed method in a reinforcement learning framework for the efficient processing of whole slide images. Our approach has the potential to significantly improve the accuracy of histopathological image analysis, which could lead to better diagnosis, prognosis, and treatment of cancer.

Acknowledgements

I would like to express my gratitude to all those who have contributed to the completion of my thesis. First and foremost, I would like to thank my supervisors for their guidance and support throughout the thesis namely Professor Santa DI Cataldo, Dr. Francesco Ponzio, and Professor Xavier Descombe. I would like also to thank pathologist Damien Ambrosetti for his advice, guidance, and continuous support.

Table of Contents

List of Tables

List of Figures

Acronyms

AI
artificial intelligence

RCC
renal cell carcinoma

ccRCC
clear-cell RCC

pRCC
papillary RCC

DL
deep learning

ML
machine learning

CV
computer vision

NLP
natural language processing

SL
supervised learning

weakly-SL
weakly supervised learning

SSL

self-supervised learning

semi-SL

semi-supervised learning

CNN

convolutional neural network

RNN

recurrent neural network

LSTM

long short term memory

WSI

whole slide image

MSE

mean squared error

ROI

region of interest

Chapter 1

Introduction

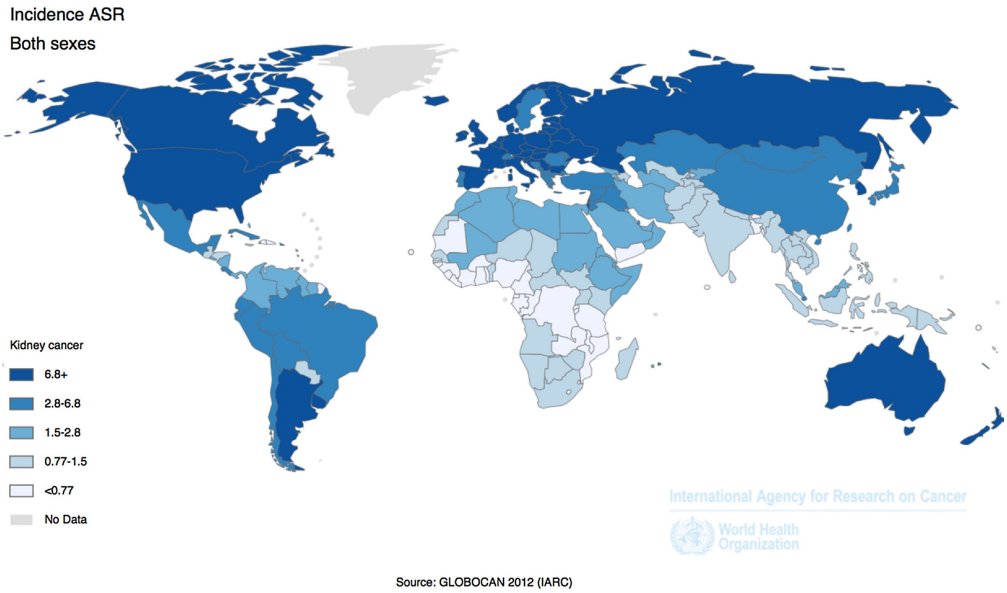
1.1 RCC

The kidneys are two bean-shaped dark reddish organs playing a significant role in our well-being by filtering blood and removing waste, along with other vital functionalities. Epithelial tissue is one of the four basic animal tissues, and the Proximal tubule epithelial cells are the most present cells in the kidney. Various diseases originate from these cells, whereas renal cell carcinoma or RCC [1, 2] is one of them. RCC is a highly malignant tumor and the most widespread type of kidney cancer accounting for 90%, this tumor isn't only the most common cancer in the kidney but also the 7^{th} most common histological type in the west (Fig 1.1a), and it is continuously increasing. Its mortality rate is considered high with respect to its incidence rate (Fig 1.1b) as this tumor is typically asymptomatic at the early stages for many patients, this leads to late diagnosis of the tumor where the curability likelihood is lower.

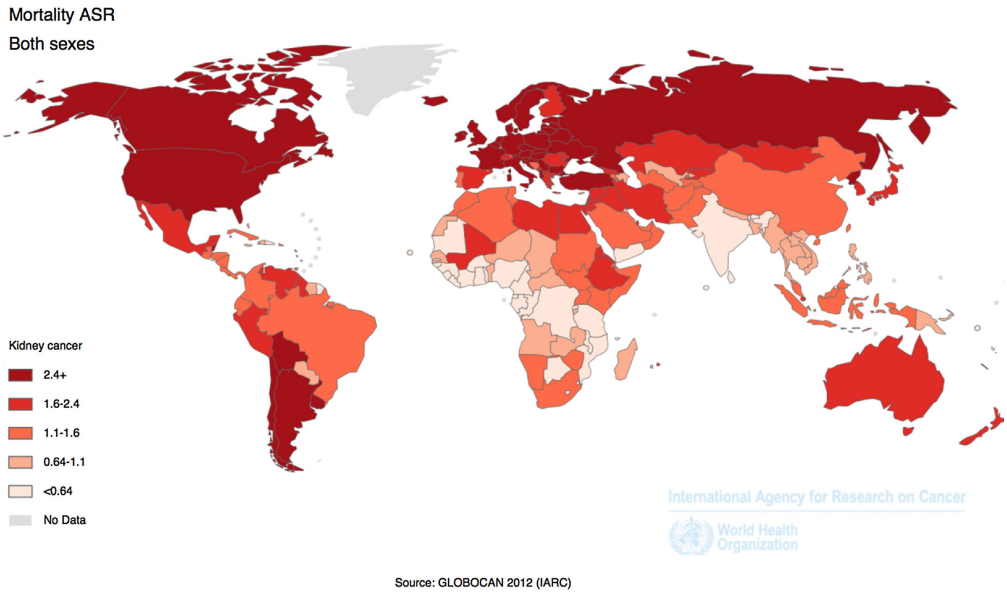
RCC can be categorized into multiple histological sub-types, mainly clear cell RCC forming 75% of RCC, papillary RCC accounting for 10%, and chromophobe accounting for 5%. Some of the other sub-types include collecting duct RCC, tubulocystic RCC, and unclassified [3]... these sub-types have different cell and structural level histological features. hence the gold standard to classify them is the microscopic visual assessment of Hematoxylin and Eosin (H&E) stained slides (WSIs) performed by pathologists. The manual diagnosis of the large WSI is both effort-requiring and time-consuming, which in some cases results in poor alignment between pathologists [4].

The correct categorization of the tumor sub-type is of major importance as prognosis and treatment approaches differ based on it and the disease stage. For instance,

medication choice differs with sub-types (refer to [5] for more information on treatment guidelines).



(a) World incidence ASRs for both sexes. Numbers are expressed per 100 000 people [6, 7].



(b) World mortality ASRs for both sexes. Numbers are expressed per 100 000 people [6, 7].

Figure 1.1: kidney cancer worldwide statistics

1.2 Deep Learning

In recent years, DL networks have seen their breakthrough with the introduction of convolutional layers and CNN, specifically, with the establishment of Alexnet [8] in 2012 and its major classification performance boost on image-net [9] 1000 class challenge. Since then, new improvements and advances in CNN technologies are continuously and frequently coming to light as more and more research is carried out in the field. After Alexnet in 2012 which utilized convolutional, fully connected, and dropout layers, deeper networks like VGG-16, and VGG-19 [10] were developed along with the inception model family starting with Google-net [11]. However, the next major step was the invention of residual connection with Resnet [12] in 2016, this mechanism opened the possibility for training and developing extraordinarily deeper networks. Resnet at the time surpassed the best accuracy records in all CV task challenges. The majority of later networks adopted the powerful residual connection within its architecture with some of them adopting other exceptional layers. For instance, a powerful normalization layer is batch normalization [13] that allows faster training by minimizing the covariate shift with its alternatives: instance [14] and layer [15] normalization... The literature of CNN is full of technologies and architecture, whereas new cutting-edge CNN networks are still being developed.

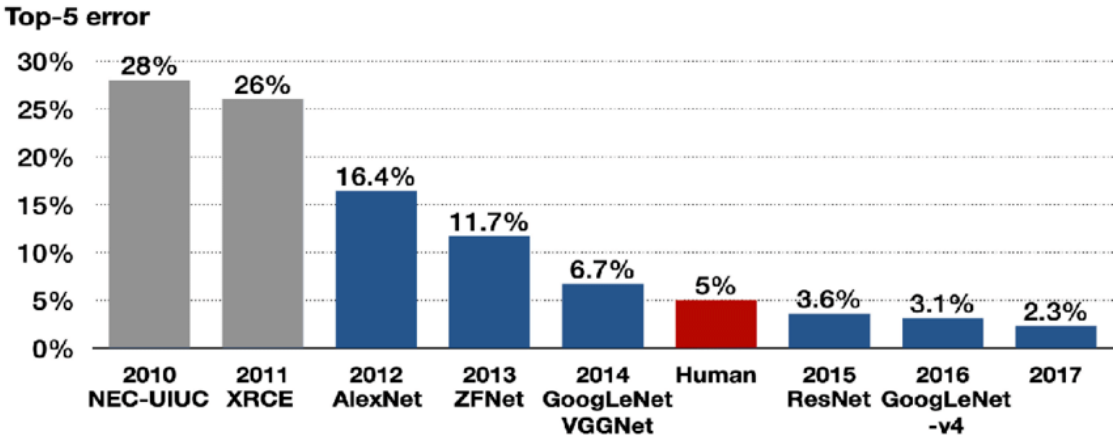


Figure 1.2: Algorithms that won the ImageNet Challenge in 2010–2017. The top-5 error refers to the probability that all top-5 classifications proposed by the algorithm for the image are wrong. The algorithms in blue are CNN. [16].

On the other hand, NLP tasks have relied primarily on RNN, a recurrent cell is a layer that accepts two inputs. The first is the previous layer output or the model input at instant t and the second is the hidden state at instant t-1 which resemble the memory of the model. This network had two major problems, an exploding

gradient and a vanishing one (short memory), a technique known as gradient clipping can be used to overcome the exploding gradient while an alternative cell called LSTM [17] was designed as a solution for vanishing gradient. However, the model was still incapable of recognizing the correlation between distant inputs and it didn't work well with lengthy sequences until the introduction of attention and self-attention [18, 19]. These mechanisms revolutionized NLP networks by utilizing the transformer blocks [19], a special block built upon self-attention. Modern models are formed following an encoder-decoder scheme to tackle sequence-to-sequence problems achieving remarkable results on a variety of tasks including summarization, information retrieval, translation, question answering... Due to their great success, attention layers have been fused with CNN to form even more sophisticated networks with millions and sometimes billions of parameters to train, leading to data-hungry architectures.

1.3 Self Supervision

As mentioned earlier, the great success of DL gave birth to immensely deep and complex architectures, requiring a vast amount of annotated samples to feasibly train them in an SL scheme for handling various specific use cases. Transfer learning is a technique that suggests the usage of previously trained networks (weights transfer) by fine-tuning them on downstream jobs without the need for huge data sets. The application of this method has indeed greatly decreased the necessity of large annotations, yet, it has one condition, the availability of sizable, general, and well-designed data sets. Luckily, image-net [9] is considered one having such attributes, still, the best practices and required labels are strongly related to domain and task similarity between the end-task and image-net challenge (see Table 1.1 for more details). The fast adaptation of DL in diverse fields has pushed new challenges on Transfer learning due to domain scarcity, hence, alternative learning paradigms such as semi-supervised, unsupervised, and weakly supervised learning were used to compromise transfer learning's shortcomings.

	similar domains	**disparate domains**
abundant annotations	all layers can be fine tuned for exceptional performance	fine tune all the network since training the last layers and fine tuning the backbone head may not work well
few annotations	train the end-part of the model and fine-tune a few layers	the model will probably fail, training the end-part can be tested

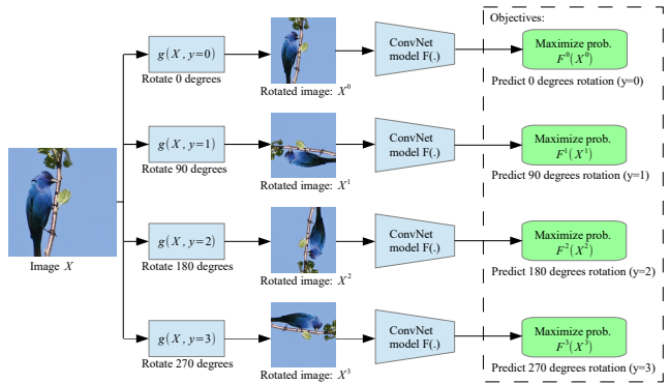
Table 1.1: Best practices according to data abundance and domains similarity

SSL is one of these paradigms, it encompasses the usage of artificially created

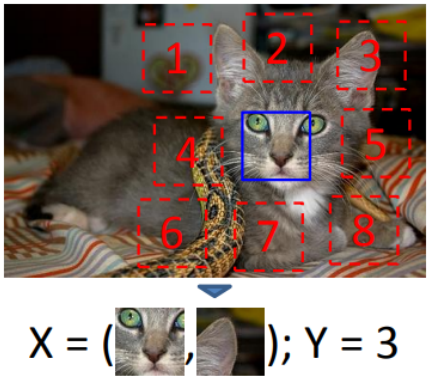
annotations using the natural structure and meta-data of the task's domain, automatically generating vast data sets to be used as a pre-training step for the network. The network is fine-tuned on the downstream job later on. SSL uses only unlabeled data similar to unsupervised learning, however, it is different at the learning level (objectives). Where unsupervised learning tries mainly to extract similarities among the samples, assigning them into clusters, usually without using feedback step (back-propagation). Though, SSL utilize artificial labels to optimize an artificially connected objective through a supervised loss and scheme, it operates similarly to how humans precept and solve a puzzle. The following are some of the SSL algorithms found in the literature:

- **Predicting image rotations:** a pretext task aimed at estimating geometric transformation applied on input images, in particular predicting rotation angles [20] (Fig 1.3a).
- **Patch localization:** a patch is randomly selected from an input image respecting specific rules, next, one out of eight patches surrounding it is selected randomly and both patches are fed to the network, the objective is to predict their relative position [21] (Fig 1.3b).
- **Image colourization:** an image is processed by turning it to gray-scale, this gray-scale input is fed to an auto-encoder architecture, and the objective is to output a colourized version that represents the original image. An MSE loss can be used in this case [22, 23] (Fig 1.3d).
- **Jigsaw puzzle:** an image or a crop of it is divided in $X \times X$ grid and a tile is taken from each cell (height and width can be the same as the cell ones), those tiles are then randomly permuted and fed to the model, the task is to predict the performed permutation [24] (Fig 1.3c).

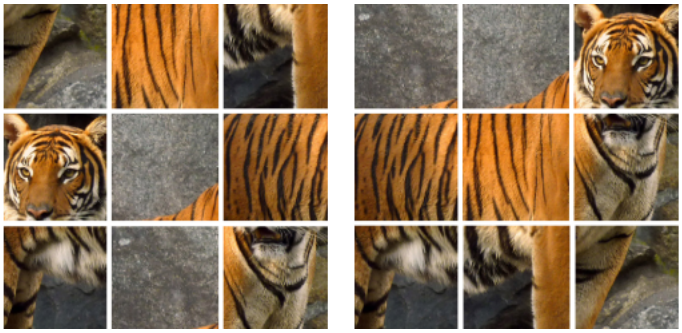
all of the above are SSL algorithms that can be adopted for CV tasks. Actually, the discriminator part of a generative adversarial network GAN [25] can be considered as a model trained following SSL paradigm and can be later used on downstream problems. There are other SSL techniques that were developed for sequence problems as videos and language, following the continuous natural order of sequences such as next word prediction [26] and shuffle & learn [27].



(a) Predicting image rotations [20]



(b) Patch localisation [21]



(c) Jigsaw puzzle [24]

(d) Image colourization [23]

Figure 1.3: Some SSL algorithms. In (a) the model is trained to classify 4 different rotation angles. In (b) the model classifies 8 relative positions, the gap between the patches is to avoid trivial solutions. (c) shows a possible permutation for the jigsaw puzzle, note that not all possible permutations should be taken into account so as not to over-complicate the task, an adequate subset can be used instead. Finally (d) presents grayscale images used for colourization and their ground truths, the encoder architecture of the trained auto-encoder is the part used for the end task.

Chapter 2

Related Works

Whole slide scanners are machines that allow digitized images of whole tissue slides to be quickly produced even at microscopic resolution. They were introduced in the 1990s and they are the main reason for the current interest in applying ML and DL methods in Histopathology. Before the wide adaptation of DL in the medical field, traditional ML was investigated for Histopathology focusing on feature extraction both on the object and spatial levels (handcrafted and domain-inspired), dimensionality reduction, and Manifold Learning... Furthermore, their applications were directed onto segmentation and detection of histologic primitives, tissue classification, grading, and precision medicine (papers [28, 29] carry attentive studies on ML in digital pathology). However, the ability of CNN to learn discriminative features directly from the raw data not requiring specialist input from pathologists has fueled the explosion of interest in deep learning applied to histopathology [30]. This chapter is divided into 2 sections, the first will tackle the advancement of deep learning generally in the field with a special focus on tumor classification, and the second will address the developed self-supervision algorithms in a histopathologic context.

2.1 Deep Learning in Histopathology

The earliest adaptation of DL in histopathology took place with *Ciresan et al.* [31], by applying CNN-based pixel-wise classifications to detect mitosis in H&E stained breast cancer images, they were able to outperform all other contestant models at the ICPR 2012 mitosis detection competition. their work was overthrown later on with the introduction of faster-RCNN [32] and its adaptation in the field [33]. Nonetheless, the work of [31] was indeed the key to utilizing DL to revolutionize digital histopathology. One major aspect to point out is the main strategies addressing the different tasks present in this domain. The data representing

histopathology data sets are normally WSI, they are large images that are hundreds of megabytes or a few gigabytes in size which make feeding them directly to deep networks impracticable (for more information regarding WSI images, please refer to section 3.1). Instead, they are divided into patches before using them for training. Thus, dividing data sets into two categories: heavily annotated with regional or patch annotations, and weakly annotated having only WSIs annotations. Therefore, the strategies adopted are based on weakly-SL or SL.

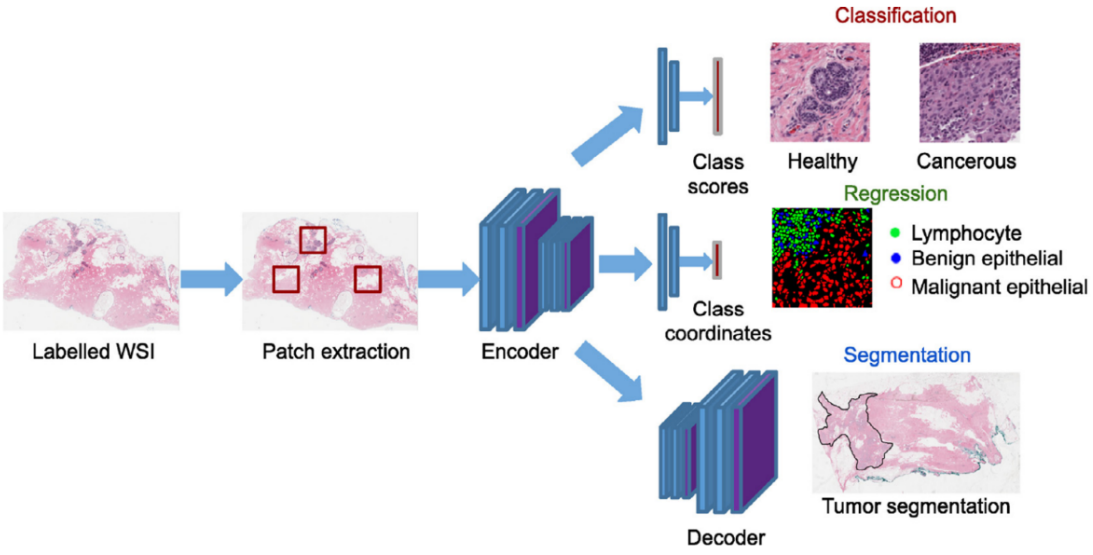
2.1.1 Supervised models in Histopathology

Histology-supervised tasks are divided into three main categories: classification, regression, and segmentation (Fig 2.1). These categories shape the used model architecture and training objectives. Histological segmentation focuses on identifying and contouring histological primitives like cells, glands, and nuclei... Early works utilized CNNs to perform pixel-wise classification, to put it another way, segmenting an image by classifying it pixel by pixel. *Kumar et al.* [34] utilized a CNN to identify nuclei, they fed the network 51×51 windows around the pixels to classify them as nuclei, background, or boundary. Performing pixel-wise classification is highly inefficient, hence segmentation techniques started adapting architectures such as fully convolution networks [35] (FCN) and UNet [36], these architectures can segment an entire image in one pass. *Seth et al.* [37] used several UNets to segment DCIS (an early form of breast cancer), while the winner of GlaS challenge [38] held at MCCAI 2015 used an architecture inspired by FCN to segment glands. Recently, novel approaches started exploiting attention mechanisms for histological segmentation [39, 40].

Similarly, the regression task objective is to localize certain objects such as nuclei or cells, they differ with segmentation in terms of their attention to separate overlapped ones. Deep regression models proposed in the histological literature are based on CNN or fully connected networks, thus, exploiting simple architectures. To leverage their performances, some authors tried to modify the output of the network to include distance constraints or a voting mechanism in the learning process [30].

Finally, histological classification with Deep networks has been used as an intermediate task for solving more complex end-task such as the segmentation of nuclei, cells, and glands... Nonetheless, its fundamental functionalities are broader, since classification's main characteristic is identifying discriminative patterns in the data to distinguish among different classes. Tumor identification and subtyping are one of the tasks requiring the extraction of high-level discriminative features at cell and tissue levels. while tumor identification stands for verifying its presence,

subtyping means differentiating between a set of tumor subtypes. CNNs are the gold standard for histopathology patch classification, they have been used in a variety of tumor recognition tasks including lung adenocarcinoma, renal cell carcinoma, breast cancer, and prostate cancer... In [41], the authors used Resnet-18 [12] as their backbone to classify patches into 5 categories: lepidic, Acina, papillary, micropapillary, and solid. They used this division to assign a grade to the tumor later on, achieving an agreement of 66.6% with three pathologists. While [42] adopted a 3 level-hierarchy by implementing VGG-inspired [10] networks at levels 1 and 2, the network at the first level handles dividing patches into epithelium, stroma, and fat. The second Network operates on stroma patches by identifying tumor-associated stroma. While the last network takes 8 feature maps according to a selection procedure from network two and uses them to classify the WSI into invasive cancer vs benign/normal. *Ponzio et al.* [43] suggested exploiting 'uncertain' models for histological data cleaning, they deployed their model before the classifier and proved a notable boost in performance.



The three novels SSL methods

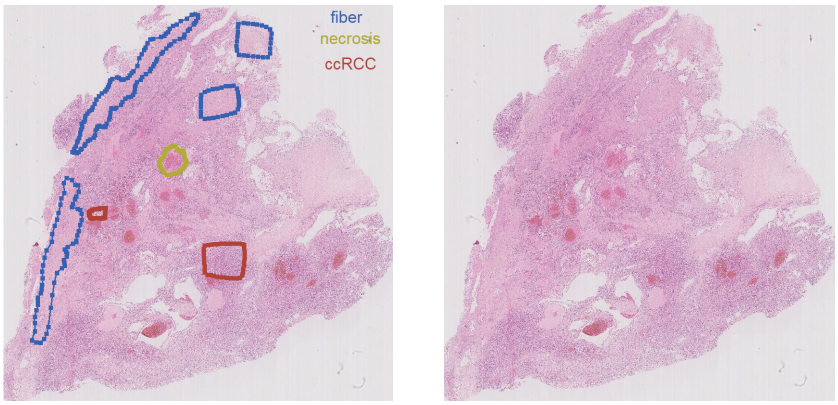
Figure 2.1: Histopathology supervised tasks [30].

Tabibu et al. [4] focused on Deep Learning for RCC recognition and subtyping. They implemented 2 CNNs, one for identifying cancerous patches and the other handles classifying them as ccRCC, pRCC, and Chromophobe. However, they argued that to overcome data imbalance, it is better to use the CNN as a feature extractor followed by a DAG-SVM [44, 45] classifier. *Xiao et al.* [46] compared deep learning networks with a combination of handcrafted features on ccRCC vs pRCC classification. To minimize annotation cost. [47] replaced full annotations

with minimal point ones where instead of contouring cancer's regions, they required only identifying them with a few points. They adopted three binary classifiers where each of them discriminates one subtype vs non-tumor. *Khoshdeli et al.* [48] demonstrated that a deep model CNN outperforms a shallow model CNN when differentiating low-grade granular tumors from high-grade clear cell RCC [49].

2.1.2 Weakly Supervised models in Histopathology

Weak supervision entails the usage of lower-quality labels or annotations at a higher abstraction level. These two types of labels are low-cost, fast, and efficient. Allowing an easy and fast building of large data sets. The first type relies on noisy or non-expert annotations, while the second is task-dependent, for example, using solely image class labels to perform object segmentation. In a histological context, exploring Weak supervision is recommended as coarse-grained information is often readily available as image-level labels, e.g., cancer or non-cancer. On the contrary, pixel-level annotations are difficult to obtain. It dramatically reduces the annotation burden on pathologists [30].



(a) **(b)**

Figure 2.2: In (a) supervised learning requires identifying tumorous and normal regions while in (b) weak supervision uses only abstracted label eg. the WSI label (ccRCC).

multiple-instance learning (MIL) is a type of supervised learning. Instead of receiving a set of instances that are individually labeled, the learner receives a set of labeled bags, each containing many instances. In the simple case of multiple-instance binary classification, a bag may be labeled negative if all the instances in it are negative. On the other hand, a bag is labeled positive if there is at least one positive instance in it. for instance, if our task is to identify the presence of a cat. Instead of processing an image of the class cat as a whole, the learner processes it

as a set of tiles. Some of these tiles will have elements of a cat in them and the rest will not, thus, creating two sets of positive and negative instances with only the whole image label available. The learner should be able to identify the image label by the existence or absence of positive instances. From a collection of labeled bags, the learner tries to either induce a concept that will label individual instances correctly or learn how to label bags without inducing the concept.

MIL was introduced as a supervised task for normal-size natural images. Nonetheless, it fits weak supervision for histological data in a way that seems as if it was created specifically to address this problem in the first place. Hence, most of the literature addressing weak supervision in Histopathology used MIL [50, 51]. *Hou et al.* [52] integrated a two-level training approach based on an expectation-maximization MIL method. They proceeded with an initial classification of all instances, then they applied a thresholding technique to filter out non-discriminative ones. The remaining serve as inputs to the second level. At level two, after obtaining patch-level prediction, instead of applying max-pooling to obtain the WSI label. They feed the predictions to a logistic regression or a support vector machine model to fuse them. [52] applied this methodology to glioma and Non-Small-Cell Lung Carcinoma subtyping. *Jia et al.* [53] adopted an FCN-like MIL model for cancerous region segmentation within a weak supervision framework. They argued that despite the model working well, it tends to assign positive labels, thus the positive instances predicted by the algorithm tend to progressively outgrow the true cancerous regions. To counter this effect, they introduced an area constraint that limits the positive region's expansion by exploiting an additional rough estimate of the cancer area. This estimate should be provided by pathologists along the labels. *Huang et al.* proposed an architecture (CELNet) that detects the presence of cancerous regions and localizes them using weakly supervised data. *Ilse et al.* [54] addresses the issue of pooling in MIL approaches. The pooling operation following the processing of instances is normally predefined and not learnable. This poses an issue when the pooling is done for the embedding of the instances instead of their predictions. To alleviate this issue, [54] suggested using an attention mechanism.

Chen et al. [55] proceeded following [54] by exploiting vision transformers for histopathology. However, due to the large number of patches extracted from the WSI. It is impracticable to use them all in one sequence as neither the memory nor speed of current GPUs can Handel processing them through a self-attention mechanism [19]. The authors introduced a hierarchical scheme to overcome this shortcoming. They divided the WSI into a hierarchy of patches, where a patch at level n is defined by smaller patches at level n-1 (level 0 is the full resolution). The first transformer (Fig 2.3 embeds the patches at the lowest level and pools those embeddings into a single one. This embedding serves as the representation of

the corresponding patch at the upper level. All other transformers proceed in the same way until obtaining a WSI-level embedding, which is used for classification. they achieve state-of-art results on BRCA subtyping, NSCLC subtyping, and RCC subtyping.

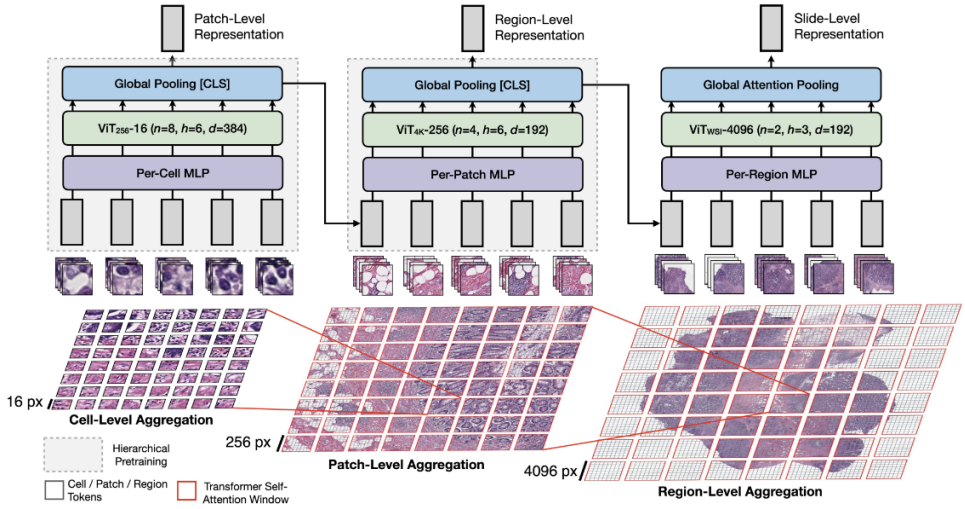


Figure 2.3: Hierarchical transformer architecture [55].

2.2 Self-Supervision in Histopathology

Self-Supervision was introduced as a solution for annotation-hungry models in their application domains. However, most of the major techniques were created and tested primarily for natural images. Therefore, using these raw techniques [20, 24, 23, 21] without any modification may not produce the desired results in other domains. For instance, in histopathology at the cellular level, objects like cells and nuclei have symmetrical shapes with respect to rotation [20]. Additionally, the tissue region is homogeneous, making it difficult to allocate patches or identify their relative positions [21] using only them as the model's input. It is simply a problem with no definitive solution. While raw colorization [22, 23] can be informative if tuned properly. In H&E stained images, the hematoxyline turns the nuclei purple while the Eosin turns the rest of the tissue pink. *Koohbanani1 et al.* [56] developed three new methodologies for Histological data (Fig 2.4) and compared them with standard SSL approaches on cancer classification datasets. They developed a magnification prediction task, where a CNN is presented with patches taken at different magnification levels and must predict their corresponding level. The second is a task inspired by the jigsaw puzzle [24], called jigmag, where

the network should identify the magnification ordering scheme of a sequence of four images. Images are taken out of four different levels, stacked randomly in a square, and fed to the model which assigns them to one of twenty-four possible outputs. Lastly, they modified image colorization to make it more suitable for histopathology. The modified version focuses on identifying nuclei by segmenting hematoxylin channel pixels. The model process the RGB image and outputs a binary mask. They compared their three pretext tasks to predefined ones such as rotation and flipping prediction, autoencoder, and generative models on three cancer classification datasets. They show that the newly histology-specific tasks perform better than the standard ones by a significant margin, especially in low annotation regimes. They also show remarkable results for histology domain adaptation problems.

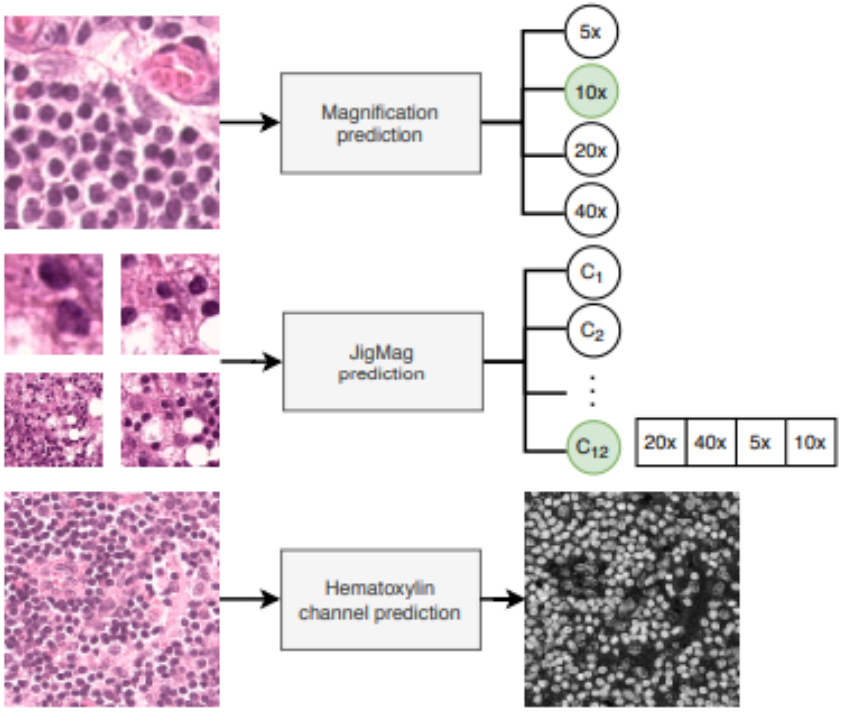
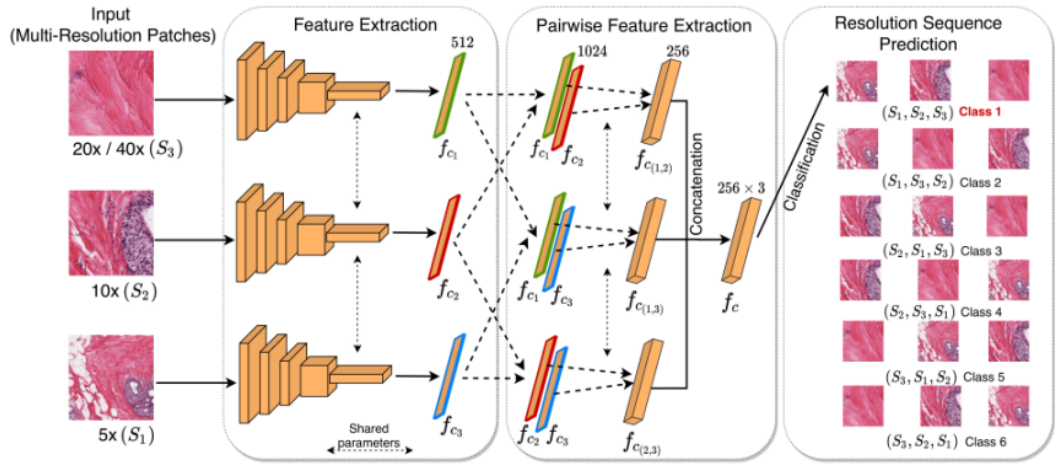


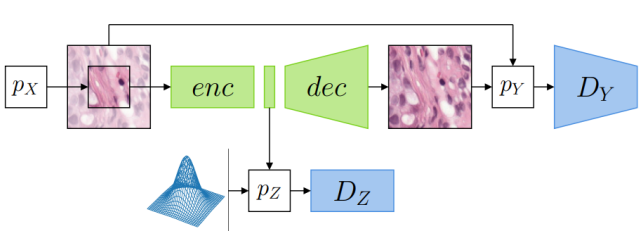
Figure 2.4: The three novel SSL methods developed by [56]

Srinidhi et al. [57] proposed a three-phase training scheme that integrates SSL, semi-SL, and consistency learning. Their self-supervised task, called resolution sequence prediction (RSP Fig 2.5a), is a variation of the magnification puzzle task (jigmag) [56]. The main differences lie in two aspects: first, the inputs are processed as a sequence of three independent images (not stacked), transforming them into latent vectors, which are then fused to produce a prediction, while jigmag stacks

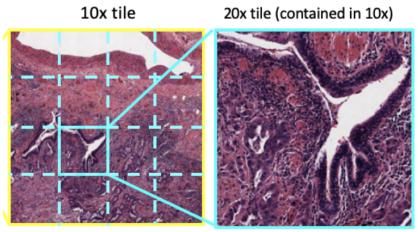
all four images in a single input. Second, for RSP, they ensure every patch is taken from the center of the lower resolution one. After pretraining the network with RSP, the authors fine-tune it on the main task with the available annotations. In the last phase, they combine consistency learning [58] and teacher-student semi-SL method to calculate a consistency loss mixed with supervised cross-entropy loss to train the model. *Boyd et al.* [59] deployed an improved version of an unsupervised learning task known as visual field expansion and employed it as an SSL pretext task. Image expansion is a problem that addresses creating a wider view of an existing image. For instance, by taking a crop of an initial image, an autoencoder-inspired network can be used to expand the crop and form the entire image. The authors' model leverage the adversarial autoencoder framework [60] (Fig 2.5b), and their objective contains three losses. A reconstruction loss and two discriminator losses. The first discriminator loss is applied at the latent space level to ensure the contextual representation conforms to a predefined distribution Pz. While the second is at the generated image level, where the discriminator tries to distinguish between real and expanded images. Most of the methods discussed in the literature focus on the importance of low and high magnifications and the various features available at each level. However, they do not take into account the interconnection between the patches taken from different levels across their tasks and within their objectives. A recent task in literature by *Ding et al.* [61] tackled this point by designing a pretext task where the model must understand if a magnified patch lies inside or outside of a zoomed-out one. They found that this simple task outperforms magnification predictions and other techniques including transfer learning. They established their experiments on LUAD subtyping where their method outperforms Transfer learning for acinar, papillary, and solid. While falling short for lepidic, and nontumor. Our methodology takes a similar approach as it is based on the concept of inter-level connectivity.



(a) RSP [57]



(b) adversarial autoencoder [59]



(c) [61]

Figure 2.5: (a) shows the overall pipeline and architecture of the RSP model and the fusion scheme utilized. In (b), the network used for image expansion is shown, it has encoder and decoder networks (enc, dec), a discriminator in the latent space (Dz), and another one at the image level (Dy). While (c) shows an example of two patches of the pretext task presented by *Ding et al.*, in this case, the zoomed-in patch lies inside the zoomed-out one.

Chapter 3

Data

In this work, we are addressing a histopathological task by utilizing histology-specific data, namely WSI. This data type has unique characteristics that need to be considered when performing analysis and interpretation. To provide a comprehensive understanding, we dedicate this chapter to introducing WSI, including its features and the tools available for managing and processing it. Additionally, we present our dataset along with relevant statistics to provide further context. The first section focuses on WSI, while the second section provides an overview of our dataset.

3.1 WSI

3.1.1 From Glass Slides to WSI

The traditional way of performing diagnoses in pathology relied heavily on the examination of glass slides. These slides are thin pieces of glass on which thin sections of tissue samples are mounted for microscopic examination, and are typically stained with various dyes to highlight specific structures. This allows pathologists to make a diagnosis based on the tissue's appearance. However, this method has several shortcomings, including the difficulty of sharing and storing large numbers of slides, and the potential for damage or loss over time. Digitalization of these glass slides helps overcome these challenges by reducing the physical space required for storage and preserving a backup of the slides in case of damage. Additionally, it opens up the possibility of sharing data across centers and among pathologists, enabling more effective collaboration. Digitalization also provides a more organized way of storing multiple slides belonging to the same patient, along with notes and observations. [62] contains a comprehensive study comparing the use of whole-slide digital images versus traditional glass slides for the detection of common

microscopic features, based on metrics such as time and the specific elements detected by different pathologists.

3.1.2 Aquisition

Whole slide imaging is a procedure where the physical glass slides are transformed into high-resolution digital images. Using a digital microscope or slide scanner, the acquisition procedure goes as follows:

- **Preparation:** The tissue sample is prepared and mounted onto a glass slide.
- **Scanning:** The glass slide is placed on the scanner bed and scanned in a high-resolution manner. The scanner captures multiple images of the tissue at different magnifications, depending on the required level of detail.
- **Stitching:** The multiple images captured are stitched together by software to form a single, high-resolution image of the entire tissue section.
- **Quality control:** The acquired image is then processed and quality-checked to ensure that the image is of adequate quality for analysis and interpretation (Fig 3.1).

Once the above procedure is completed, the images are stored in one of many available formats such as TIFF, TIF, SVS, JPEG, etc... Each one of them provides different advantages in terms of quality, compression, fast management, and storage size...

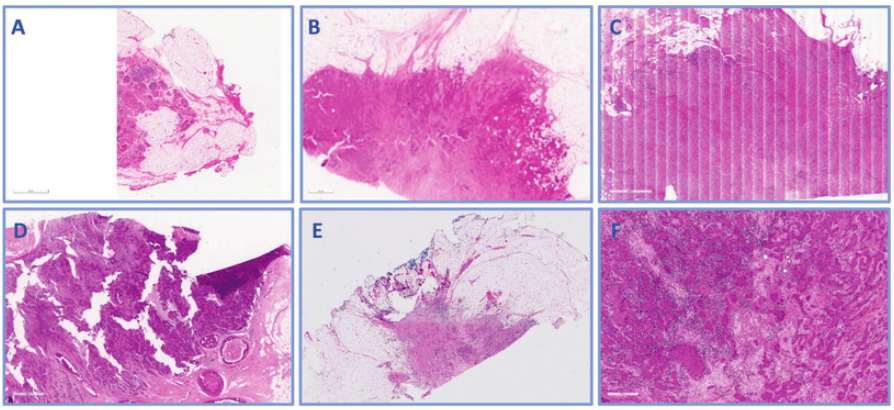


Figure 3.1: Examples of failed WSIs considering quality control. A) Incomplete slide scanning, B) Out of Focus image, C) Improper line stitching. D) Thick sections with tissue cracking and folding, E) Uneven H&E Stain distribution, F) Air bubbles on slide [63].

3.1.3 WSI Nature

WSIs are large digital images, often consisting of tens or hundreds of thousands of pixels. A typical image scanned at a magnification of $\times 40$ has a resolution of around 0.25 μm per pixel, resulting in a disk space requirement of approximately 48 megabytes (without compression) per 1-mm^2 [64]. WSI can be described as a pyramid-like structure (Fig 3.2), with the base being the highest resolution image, accounting for the largest number of pixels. As we move up the pyramid, intermediate representations of the tissue at different magnifications are obtained, with increased context (tissue architecture) at the higher levels, and more details (nuclei and cell information) at the lower levels (Fig 3.3). At the top of the pyramid lies a full representation of the slide, known as the thumbnail. Inter-level maps can be obtained using pooling operations, such as max pooling if needed. Due to the continuous zooming in and out during analysis, WSI is normally accessed as patches from different resolutions and is never accessed as a full slide. As a result, vendors and visualization software store it in partitions and only read the necessary ones. In the next section, we will showcase software and libraries that can be used to manipulate these slides.

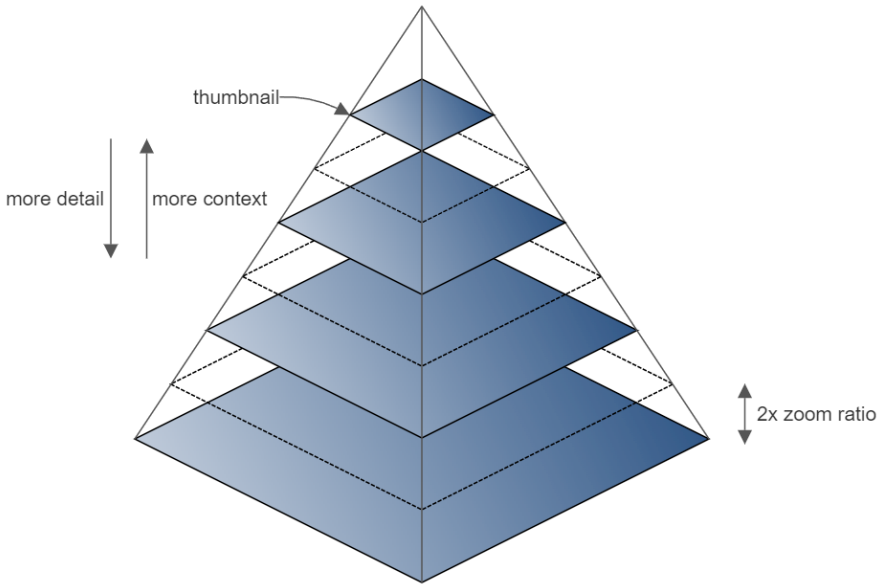


Figure 3.2: Pyramidal structure of WSI. At the top we have the lowest-resolution image showing all the tissue, this image is referred to as a "thumbnail". Going down we have more details while moving upward we get more context. The base of the pyramid is the highest resolution map of the WSI taken at maximum magnification.

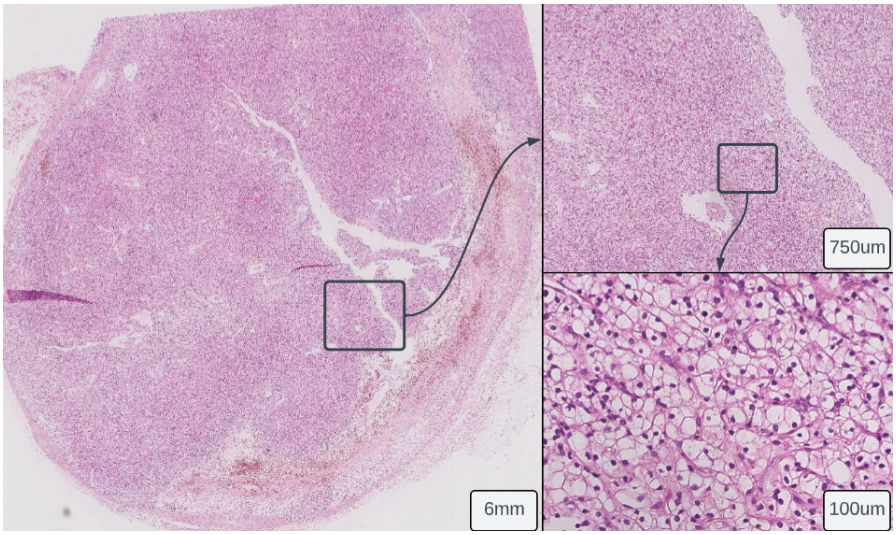


Figure 3.3: multi-resolution patches

3.1.4 Software & Libraries

Whole slide imaging offers numerous advantages, resulting in the development of various software programs for viewing, managing, and analyzing these large images. ASAP (Automated Slide Analysis Platform) is one such powerful and flexible tool that is designed specifically for this purpose. ASAP helps with visualization, annotation of regions of interest ROIs, and management of WSI. The annotations are stored in organized XML files, which makes it easier to read and modify them for downstream ML and DL tasks. Additionally, ASAP provides some utilities for nuclei detection and color deconvolution. Some other platforms include QuPath, ImageScop, Aperio eSlide Manager, and Pathomation.

ASAP is built on the open-source library, `Openslide`, which provides methods and functions for reading and manipulating WSI. The most noteworthy ones include the The `read_region` function and the `Deep_Zoom_generator`. The `read_region` function helps to extract patches of specific sizes from any magnification level in the WSI, while the `Deep_Zoom_generator` divides the WSI into a grid of tiles.

3.2 Our Data-Set

The objective of this study is to classify different subtypes of RCC, namely ccRCC, pRCC, chromophore, and oncocytoma (Fig 3.4). Hence, the dataset consists of patients diagnosed with one of the mentioned subtypes stored in separate folders based on the classes. Each file is identified by a patient identifier and a WSI

identifier. The slides were collected by pathologists at CHU Nice using H&E staining and annotated by former students. The dataset is heterogeneous in terms of file formats and annotations. For instance, the main format used for ccRCC and pRCC data is SCN, and ASAP was used for annotating them. The annotation process focuses on identifying homogeneous ROI for both tumor and non-tumor classes such as fiber, normal cell, and necrosis (Fig 3.4), with the homogeneity consisting of having a single class within the ROI and excluding boundary regions. On the other hand, oncocytoma and chromophobe folders contain samples with TIF and SVS formats, and their annotation was produced by cropping the tumor ROI out of the slides. As a result, no non-tumor annotations are available for them.

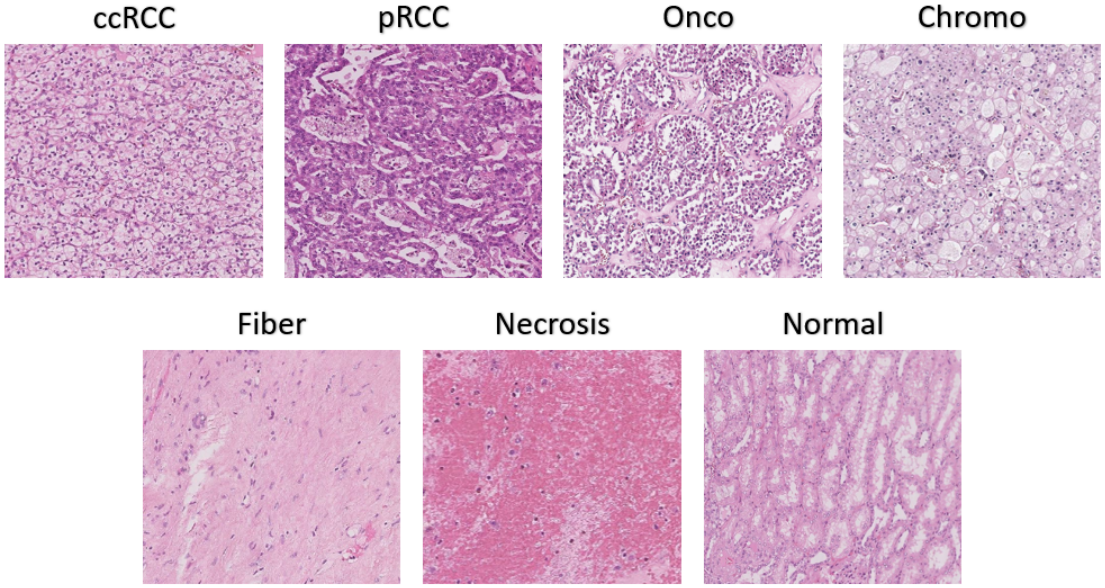


Figure 3.4: Tumor & non-tumor classes. The first row shows all RCC subtypes while the second one shows the non-tumorous regions of the tissue

The dataset contains a total of 91 patients diagnosed with one of the four RCC, as shown in the table 3.1 The data is heavily imbalanced, with the majority class being ccRCC. This imbalance is due to the actual distribution and rarity of the different subtypes, as previously mentioned in section 1.1, ccRCC is the most prevalent RCC (accounting for approximately 75%). The data was split on a patient level to ensure that the model is tested on patients it has not encountered during training, as tumor characteristics can vary between patients. Initially, the data was divided into three sets: train, test, and validation. However, due to the low abundance of patients with chromophobe and oncocytoma, the test set was unrepresentative for these two subtypes, so the initial split was adjusted. The validation set was sacrificed and replaced with a split taken at the patch level. This means that the

dataset was first split into a train and test set on a patient level, and then after processing the training data and transforming it into patches, the train set was further split into a train and validation set. Although this approach solved the previous problem, the new validation set is less representative.

Subtype	Initial-Split			Final-Split		Total
	Train	Validation	Test	Train	Test	
ccRCC	36	6	14	33	23	56
pRCC	14	2	6	15	7	22
Onco	5	1	1	3	4	7
Chromo	4	1	1	3	3	6

Table 3.1: Patients division across splits and subtypes. The table shows how patients are distributed with respect to the four subtypes and by splits' sets both the initial and final one.

Lastly, the time of detection of a tumor differs among patients due to RCC being asymptomatic cancer (as discussed in Section 1.1). As a result, the growth and size of tumors vary from patient to patient, leading to different regions of interest ROI and instances of the tumor being found across the dataset. Table 3.2 displays some statistics that will prove useful in defining our downsampling strategy later on

Statistic	ccRCC	pRCC	Fiber	Necrosis	Normal	Onco	Chromo
Nb-Slides	69	36	78	26	44	18	8
Nb-Annotations	307	141	269	95	85	___	___
avg-Annotations	4.45	3.92	3.45	3.65	1.93	___	___
max-Annotations	19	9	14	23	7	___	___
min-Annotations	1	1	1	1	1	___	___

Table 3.2: Class statistics. Nb-Annotations is the total number of annotations for the class while the remaining statistics are computed per slide

Chapter 4

Methodology

in the following sections, we delve into our approach, motivating it with the data underlying properties and structure, discussing possible formulations, and presenting a detailed overview of our implementation while including the challenges and limitations encountered. This chapter is divided into four main sections, starting with a general introduction and motivation, followed by a detailed formulation and an in-depth implementation, and ending with a brief overview of the pretext results and findings.

4.1 Introduction & Motivation

The objective of self-supervised learning is to foster the development of human-like common sense in the model through the learning and extraction of semantic and natural features from data, regardless of their discriminative nature for a specific downstream task. This makes self-supervision methods general and reusable across various domains. By harnessing the inherent characteristics of the data, a well-crafted pretext task can allow the model to comprehend and identify common patterns and redundant features. Utilizing this acquired knowledge, the model should be capable of solving the defined self-supervised task. Hence the better the task is designed, the more information can be extracted. thus, SSL research is focused on finding and designing tasks serving the aforementioned purpose.

Following our discussion, rotation prediction [20] is a task that primarily relies on geometrical non-symmetrical features and can effectively extract object contours to separate them from the background of the image. The outer geometry of the object is crucial in determining the alignment and orientation of the image (Fig 4.1). This leads to a more comprehensive understanding of the object's rotation within the image and therefore, a better performance in the task. Image colors and

inter-object fine details such as a cat's eye pupil and fur play a secondary role in solving this task. In contrast, colorization tasks [23, 22] place a higher emphasis on object color and color transitions, utilizing color information to separate objects from their backgrounds and map a grayscale image to its original form. Such separation of background and object can be observed in figure 4.1. On the other hand, localization [21] considers both geometry and color to effectively understand high-level features within an object, identify similar regions, and learn discriminative features that distinguish between image regions while fully comprehending the object's structure. For instance, even if presented with just two regions, such as a cat's right ear and left eye (as seen in Figure 4.1), the model must understand the symmetrical structure of the cat and accurately distinguish between its right and left features to effectively solve the task. In addition, a jigsaw puzzle [24] can be considered a comprehensive patch localization technique as it focuses on determining the arrangement of all patches within a crop, not just the relationship between two of them.

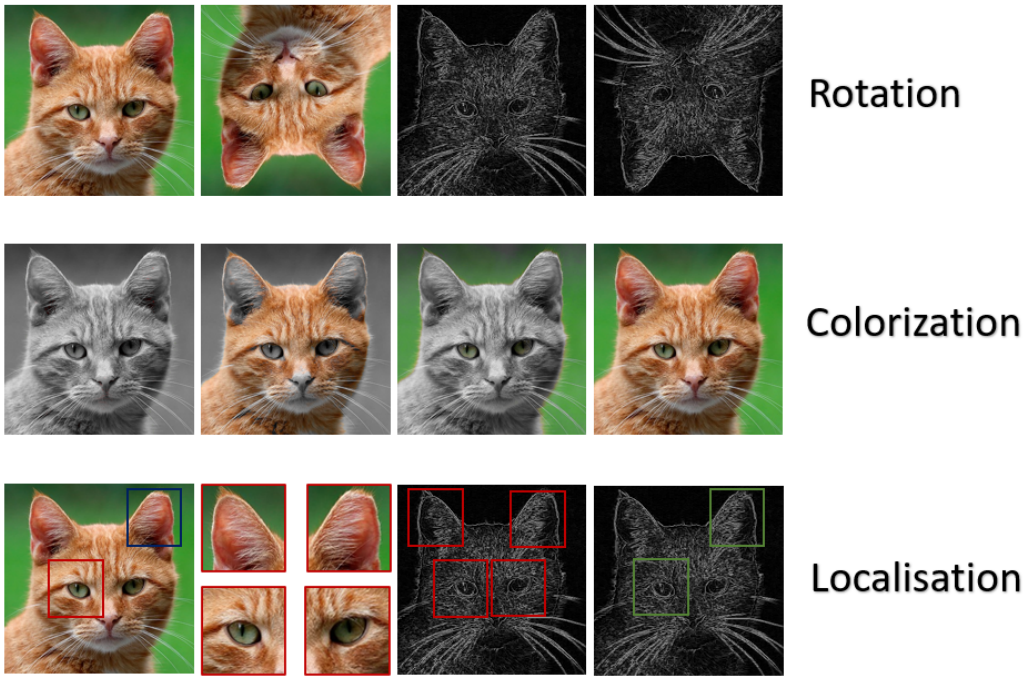


Figure 4.1: Behind pretext tasks. Useful semantics and natural features to solve the mentioned pretext tasks.

Even though these tasks were designed with care, they were intended for natural image problems and not histopathology. In digital pathology, tissues are often

homogeneous at high magnification, with simple geometries, colors, and symmetrical structures for cells and nuclei. As a result, methods relying on geometry fail to identify meaningful patterns. While colorization can still be effective to some extent due to the H&E staining that separates nuclei and tissue based on color, the intensity of color depends on the amount and type of staining material used. Hence, simple colorization cannot function optimally. This is reflected in the degraded performance of such methods, as demonstrated by [56]. The only exceptions are autoencoders and GANs, as they are more adaptable to new domains.

Therefore, the proposed self-supervised task for histopathology focused on the data nature, since designing a pretext task following only the structure is inefficient as a result of tissue homogeneity. The foundation of WSI and histopathology lies in the capturing of tissue at high resolution, providing multiple filed of view for diagnosis. This is due to the availability of various information obtained from different viewpoints (magnifications). Thus, new tasks adapted magnification-inspired puzzles, where inputs are extracted at different magnifications such as [56, 57, 61]. These tasks have shown improved performance compared to traditional methods.

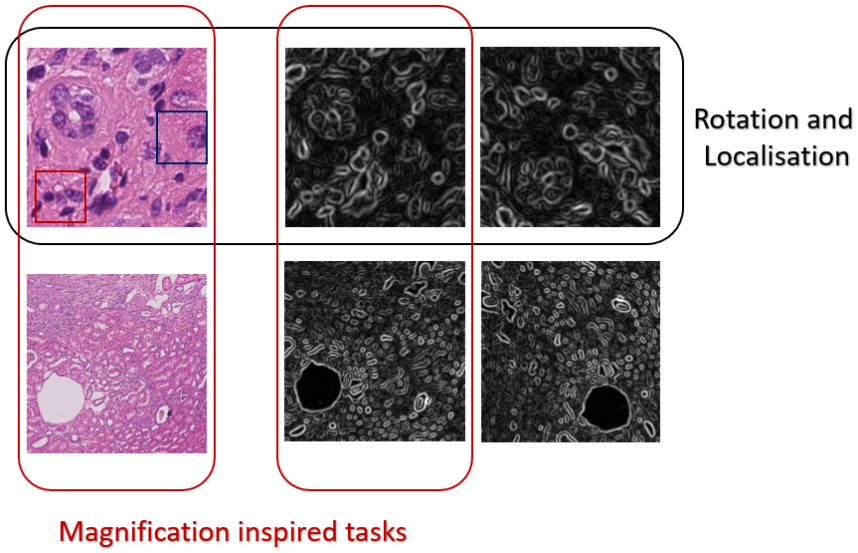


Figure 4.2: Magnification inspired tasks VS traditional SSL. The failure of rotation and localization to extract discriminative features for their respective tasks from a single patch can be attributed to symmetry and basic geometric characteristics. The magnification-inspired task takes patches from different magnifications. Although the geometry of each patch may still be simple, it varies between them. At high magnifications, the nuclei and cell structure play a dominant role, while tissue architecture becomes the key factor at lower magnifications.

Despite recent advancements in self-supervised pretext tasks for histology, such as magnification prediction and magnification ordering [56, 57], they were limited in one aspect. These approaches used high and low magnification to create their tasks, but they failed to fully leverage the relationship between high and low-resolution images. For example, they didn't consider how small features such as nuclei and cells, which are visible in high-resolution images, impact the topology when viewed at a lower resolution. Pathologists don't typically examine low and high-resolution images in isolation. They zoom in and out while keeping in mind the information they gathered from previous observations. Thus, the relationship between low and high magnification is a crucial aspect that should be taken into account.

Our approach, as well as the work of *Ding et al.* [61] explores this relationship in the underlying pretext task. *Ding et al.* used it in a simple manner to answer a binary question of whether a high-resolution patch is contained within a lower-resolution one, and their results showed that this simple task outperforms other tasks in the literature. In contrast, our method assumes the membership criteria are met and asks the model to locate the high-magnification patch within the low-magnification one. Our approach leverages the capability of patch localization methods to understand the underlying image structure by adequately adapting them to histological data. The model needs to understand both low and high-resolution features and map them in such a way that it can comprehend the overall connection between the two patches to perform the localization task accurately.

The task resembles solving a complex puzzle, where instead of only giving the pieces, a small image of the completed puzzle is also provided as a reference. In conclusion, our method takes advantage of the WSI structure through different magnifications while using a localization task that is based on the relationship between the different inputs.

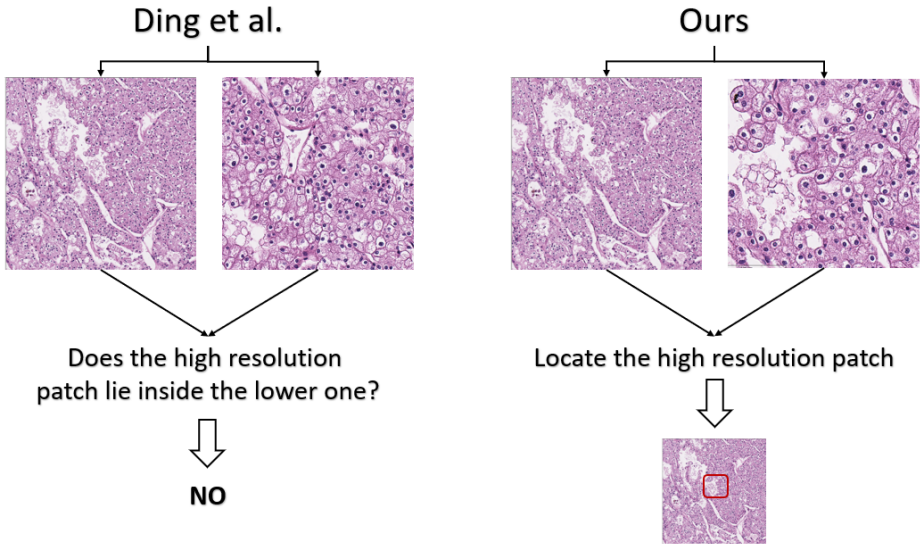


Figure 4.3: Difference between our method and *Ding et al.*

4.2 Formulation

Suppose a WSI has N magnification levels, where 0 is the max resolution level and N is the lowest. the width and height of the image at level t expressed as (W_t, H_t) are correlated with those at level $t-1$ as follows:

$$W_{t-1} = 2 \times W_t \quad \& \quad H_{t-1} = 2 \times H_t$$

Thus a patch p in t can be represented as 4 non-overlapping patches in $t-1$ having the same dimensions as p. Assume that we have 2 magnification levels x & $y \in [0 \dots N]$ such that $x < y$. Denote by P_x and P_y the sets of all possible patches extracted from the WSI at level x & y respectively with a fixed height and width. g is a mapping function from P_x to P_y which maps a subset C of P_x into an element of P_y (g map a high-resolution set into their corresponding representation in the low-resolution space).

$$g(C) = i \quad \text{where} \quad C \subset P_x \quad \& \quad i \in P_y \tag{4.1}$$

Let p_x and p_y be two patches such that $p_x \in P_x$, $p_y \in P_y$, and $p_x \in C$ where $g(C) = p_y$. In other words, p_x lies inside the magnified version of p_y. The objective of our model is to understand the location of the representation of p_x inside p_y expressed as our true label y_{true} :

$$\min_{\theta} \quad Loss(F_\theta(p_x, p_y), y_{true}) \tag{4.2}$$

where F is our model with parameters θ. So far, we have introduced a general abstract formulation that can be applied to a variety of cases for modeling the objective and handling inputs. In the following subsections, we will provide specific formulations for each of these cases.

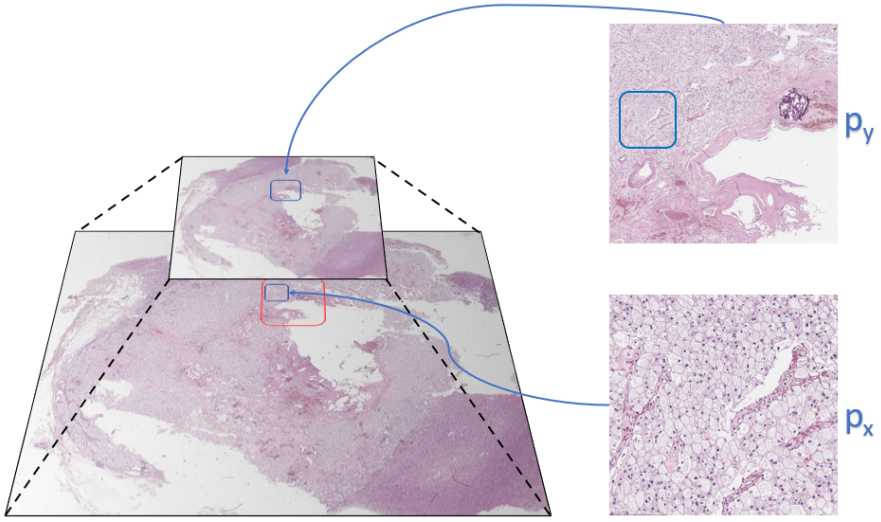


Figure 4.4: An example of p_x and p_y.

4.2.1 Fusion

First, we address the model F_θ, which comprises a convolutional neural network for feature extraction and an end part connected to our objective (we will touch on this in the next subsection). Hence, splitting F_θ into f_{w1} for the backbone and h_{w2} for the head: $F_\theta(x) = h_{w2} \circ f_{w1}(x)$.

Our task involves processing multiple inputs (p_x and p_y form one input sample). As such, we need to determine a fusion strategy. Fusion [65] is a well-studied technique in the literature, often applied to samples that include multiple data modalities (such as text and image), and involves adapting these heterogeneous domains. Leveraging information from different modalities is crucial to effectively perform many real-world tasks, as they provide diverse knowledge. Some examples of these tasks include video analysis, visual question answering, analyzing medical data, and many more. In some cases, collecting data from different modalities can even improve the performance of single-modality tasks (such as speech understanding). For example, collecting vision information can help the model observe mouth movements and improve its performance. Different types of fusion have been investigated, including input fusion, latent space fusion, and hybrid fusion.

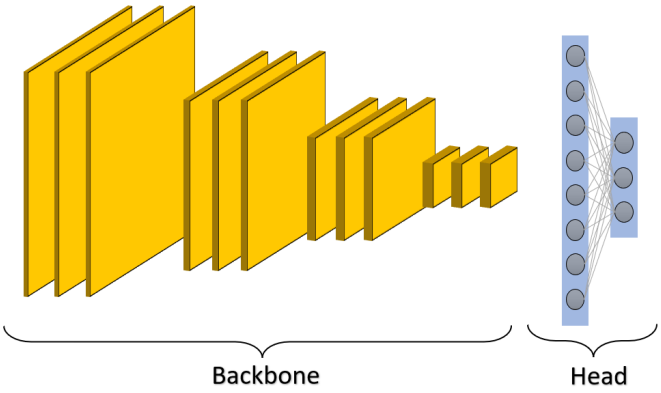


Figure 4.5: Model general structure.

Input fusion

Input fusion entails fusing the input data before feeding it to the model. This can be challenging when the data has different structures and modalities, requiring some representation adaptations before feeding it to the network. Such adaptation steps might result in information loss. However, in our case, the multiple inputs share the same nature and size, so a simple input concatenation approach may suffice to implement input fusion. Following the approach in [61], we can concatenate the images on the channel dimension, as a result, the head h_{w2} stays the same while the first convolution in our backbone f_{w1} changes to accommodate the new 6-channel inputs instead of the previous 3 RGB inputs. Another example of input fusion can be seen in [56], where they stacked four patches into a single RGB image for their Jigmag task while keeping their backbone intact.

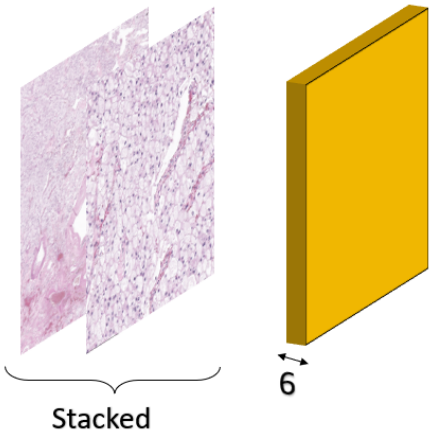


Figure 4.6: Channel concatenation.

Latent space fusion

Fusion can be performed after the extraction of latent vectors using either the same backbone network or different ones. In the case of multimodal data, adequate architectures can be used for the extraction process (eg. attention mechanism for text and convolution for images). After extraction, all inputs are represented as vectors, making fusion simple with fully connected layers or more complex functions like attention. In our case, we need our model to extract both high and low-magnification features, so we use the same backbone for both p_x and p_y feature extraction. This is similar to the approach used by [57] in their pretext-task training. In this case, f_{w1} remains unchanged, and an additional fusion layer L is added before h_{w2}. Depending on the output dimension of the fusion layer, h_{w2} may change or stay the same. It's worth noting that latent space fusion only considers high-level features and may miss correlations between low-level ones.

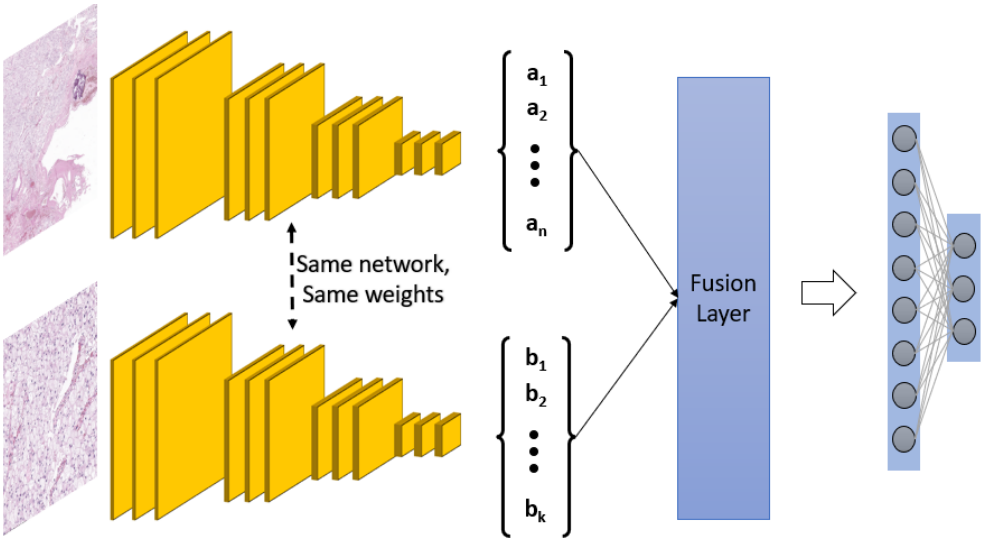


Figure 4.7: Latent space fusion.

Hybrid fusion

Hybrid fusion involves combining both input and latent space fusion methods to leverage the advantages of both and overcome their shortcomings. It can be implemented using subnetworks and aggregation operations, making it more complex than other fusion methods. However, hybrid fusion is less relevant for our method since input fusion does not require representation adaptation for the inputs. As a result, there is no loss in the process.

4.2.2 Objectives

Considering the nature of our self-supervised task, which is built upon localization, there are two possible formulations for the objective that can be used: a classification-based objective and a regression-based one. These formulations differ in the following manners:

- **Labels:** The way labels and predictions are expressed, as a consequence, the loss function also differs.
- **Flexibility:** The choice of objective can impact the flexibility in sample generation and the dataset's capacity for heterogeneity, as one objective may impose more constraints than the other.
- **Difficulty:** The task will be more challenging to the model depending on the objective used. The difficulty is also related to flexibility in terms of sample heterogeneity. The more heterogeneous the data is, the harder the task.

Moving forward, we will delve into the aforementioned objectives, providing a comprehensive overview of the points mentioned earlier.

Classification

In such an objective, p_y is partitioned into a grid of distinct and non-overlapping tiles. This grid consists of 4^n tiles, where $n = y - x$. The set representing the tiles at level x is C (equation 4.1), and one patch is sampled from it representing our p_x. y_{true} is generated using the location (i, j) of the tile represented by p_x as:

$$\begin{gathered} y_{true} = i \times 2^n + j \quad \Rightarrow \quad y \in \{0,1,2,\ldots,4^n - 1\} \\ where \quad i \,\&\, j \in \{0,1,2,\ldots,2^n - 1\} \end{gathered}$$

As the problem is now a classification task, we can use the cross-entropy loss function to measure the discrepancy between the predicted class probabilities and the actual class labels, where equation 4.2 becomes:

$$\min_{\theta} \quad \frac{1}{k}\sum_{i=1}^{k} Loss(\hat{y}_i, y_i) \quad where \quad Loss(\hat{y}_i, y_i) = -\sum_{j=1}^{c} y_{ij}\log(\hat{y}_{ij}) \tag{4.3}$$

where

- y_i: the true label y_{true} for sample i.
- k: the number of samples.
- c: the number of classes $4^n - 1$.
- $\hat{y}_i$: the model prediction $F_\theta(p_{xi}, p_{yi})$ for sample i.

In the classification formulation, the representation of labels can pose some flexibility and difficulty. We have observed that the range of y_{true} is dependent on n which is equal to $x - y$. To ensure that the range of y_{true} is maintained for our outputs, we need to fix the difference between x and y. In other words, we are free to extract patches from different levels, but the zooming difference between our two levels must remain fixed. This results in less heterogeneous data, making the task easier, as the size of tiles in the zoomed-out patch is fixed.

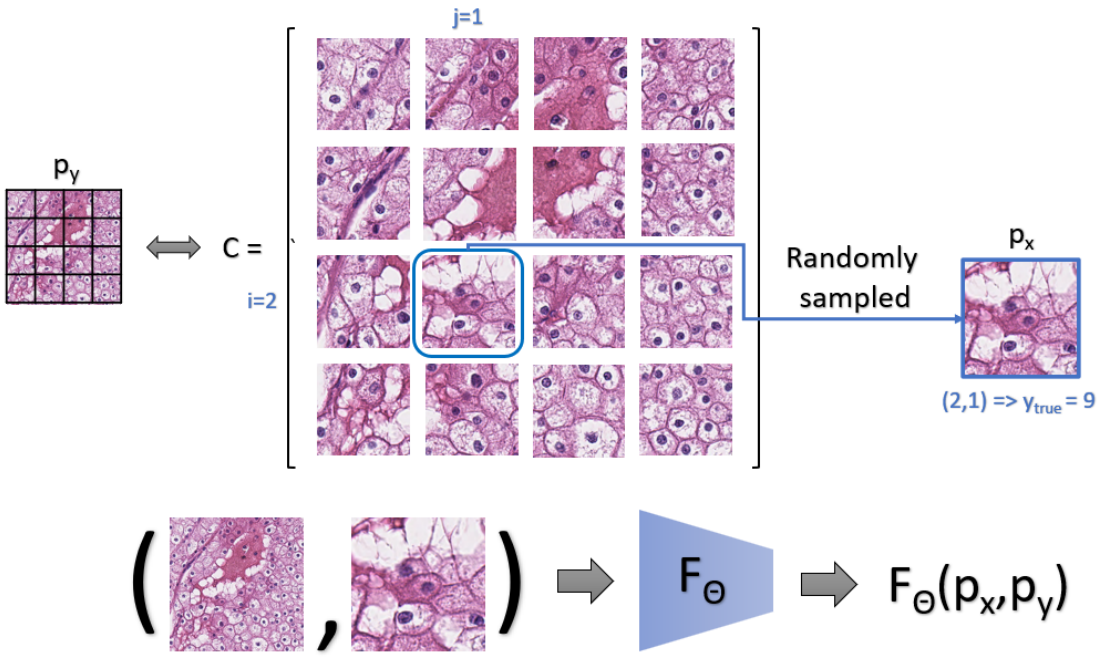


Figure 4.8: Classification formulation. In this example, n is set to 2, as a result, p_y is partitioned into a grid of 4×4 tiles. Sampling the tile at coordinate (i=2,j=1) as p_x (i and j start at 0), we obtain our label $y_{true} = 2 \times 2^2 + 1$. The output of our model is then used along with y_{true} to calculate the cross-entropy loss for the training process.

Regression

In This approach, rather than modeling p_y as a grid of tiles, we consider it as a "continuous" interval of points ranging from 0 to the width or height of the image. In such case, there are two possible methods to define C: a single element in P_x for each coordinate (i, j) representing the top left corner of a tile in p_y (one-to-one) or a neighborhood for each coordinate (many-to-one), refer to figure 4.9 for more details. Although both approaches are viable, we opt for the simpler strategy of constructing C using the one-to-one approach. As a result, C has a size of $H \times W$. When p_y and p_x share the same size, the tile representing p_x in p_y should have a size of $(\frac{w}{2^n}, \frac{h}{2^n})$. the true label y_{true} is expressed as the coordinate (i, j).

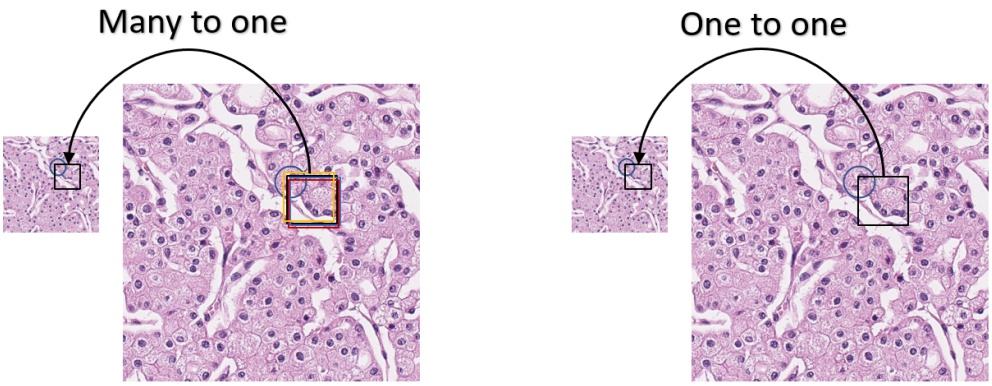


Figure 4.9: Two strategies. A set of neighboring pixels in a high-resolution patch gets mapped to a single pixel in its low-resolution version. This is where the two strategies originate. Many-to-one takes into consideration all of these pixels while one-to-one takes only one (center...).

In this formulation, utilizing the mean squared error loss may not be the optimal approach, as tissues are typically homogeneous, and the data is artificially generated. Causing the model to output predictions in the vicinity of the true label. However, for some images, there may be regions that are almost identical, resulting in incorrect predictions. To ensure a smooth training and prevent exploding gradients, it is better to use the $smooth_{L_1}$ loss, as explained in [66]. To account for predictions that are slightly inaccurate, we define a circular neighborhood around y_{true} with a radius R, and we adjust $smooth_{L_1}$ and represent equation 4.2 in the following way:

$$\min_{\theta} \quad \frac{1}{k} \sum_{i=1}^{k} \text{smooth}_{L_1}(\hat{y}_i - y_i) \tag{4.4}$$

in which

$$\text{smooth}_{L_1}(b) = \begin{cases} 0.5(\frac{b}{R})^2 & if \quad |b| < R \\ |b| - R + 0.5 & otherwise \end{cases}$$

where

- y_i: the true label y_{true} for sample i.
- k: the number of samples.
- R: is the radius, which can be considered as a hyper-parameter.
- $\hat{y}_i$: the model prediction $F_\theta(p_{xi}, p_{yi})$ for sample i.

The Regression formulation provides more flexibility, allowing patches to be extracted from different magnification levels, while also allowing the magnification gap between p_x and p_y to vary, which wasn't the case with the classification approach. The latter makes the task more challenging by introducing different tile sizes (inside p_y), making the data more diverse. Hence, it might be more challenging to successfully train the network.

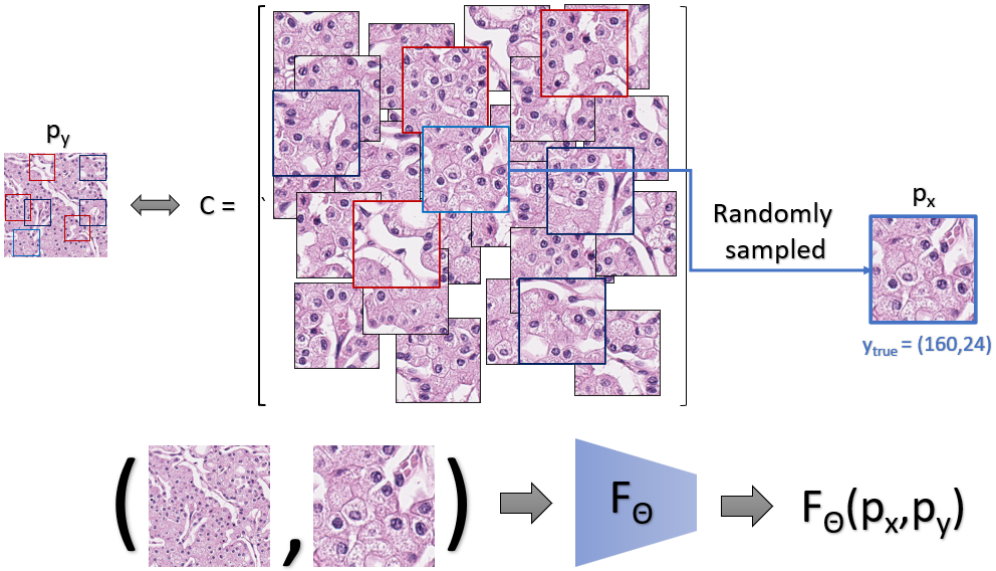


Figure 4.10: Regression formulation. The one-to-one strategy is used, and p_x is randomly sampled from C with y_{true} equal to the tile coordinate representing p_x in p_y. The patches here are all extracted with $n = 2$, however, n is not constrained to stay fixed.

4.3 Implementation

This section will provide a detailed account of our implementation of the self-supervised pretext task. This will include our choice of model architecture, the creation of artificial data, and the pre-processing pipeline. We will also provide an overview of the training process, including the challenges we encountered and the strategies we used to overcome them.

4.3.1 Model Architecture

We opted for a simple version of the pretext task for our preliminary experiment to evaluate its potential. Specifically, we chose to use the classification formulation in this stage. Another important decision we had to make was whether to use latent fusion or input fusion, both of which have been tested in previous studies [61, 57]. In our case, since our goal is to pre-train the model and encourage it to learn useful features from multiple magnification levels, we decided to use the latent space fusion approach. This method constrains the model, ensuring that it only uses high-level extracted features for the fusion process, and also allows for easy transferability to downstream tasks.

To ensure that the model doesn't rely on trivial solutions, we incorporate an augmentation layer at the input stage. the trivial solutions refer to instances where the model focuses on specific regions of the patches to solve the task, creating shortcuts for the network. For example, the model may use border information from the high-resolution patch to locate it inside the low-resolution one. Given that the task is a classification problem, using such tricks could be relatively easy, since the possible locations of the low-resolution patch are limited to a fixed number. To address this issue, we apply random augmentations to the input images. These augmentations force the model to consider the entire patch and prevent it from relying on shortcuts.

Additionally, we have included an optional dropout layer after fusing the latent vectors to mitigate over-fitting. This layer restricts the use of the fused features, but it also causes a significant slowdown in the training process. The general architecture of the network is illustrated in Fig. 4.11. Following this architecture, we will now present each section in detail:

- **Input:** The input to the network is comprised of two images, one at low magnification and one at high magnification, both of which share the same dimensions. Since the fusion occurs in the latent space, the two images are fed separately into the network. As a result, each input sample has dimensions of $(2, w, h, 3)$.

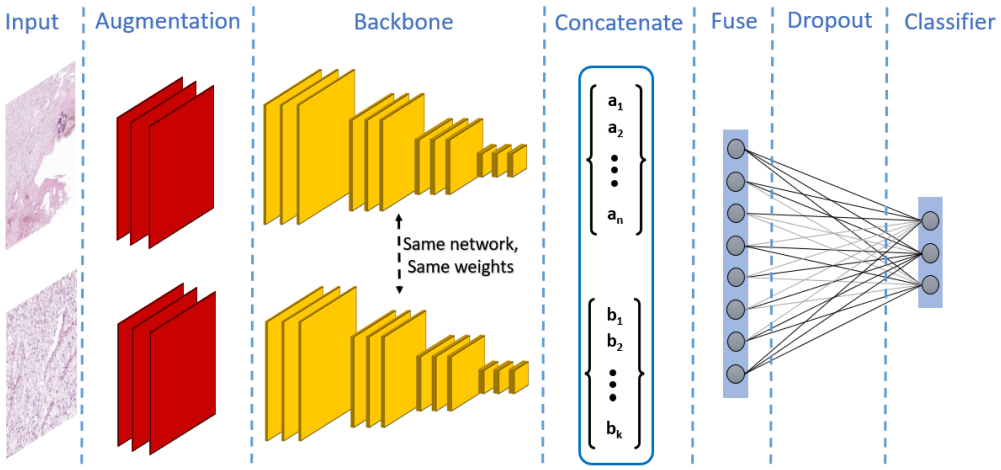


Figure 4.11: General architecture.

- **Augmentation:** As previously mentioned, augmentations are crucial for preventing the model from relying on shortcuts. To achieve this, it is important that the augmentations applied to the patches are randomized and varied. However, the applied augmentations to the low-resolution patch should not be geometrical, as this could cause confusion by altering the actual labels. On the other hand, all types of transformations can be applied to the high-resolution patch.

- **Backbone:** The feature extractor can be any CNN. Our primary choice is VGG16 [10], Additionally, we deploy a light version of Resnet-18 [12]. However, This version is only used to show some insights in our observations section 4.4. While all the significant results which we present later on are obtained using VGG16. The reason behind such a decision will be explained further in the training subsection 4.3.3.

- **Concatenate:** In this layer, we concatenate the latent vectors obtained from the feature extractor, combining them into a single tensor in preparation for the subsequent fusion layer.

- **Fuse:** To perform the fusion, we use a fully connected layer with a ReLU activation function. The number of neurons in this dense layer can be adjusted as a hyperparameter.

- **Dropout:** This layer is implemented to prevent the overfitting of the fused features. Its usefulness is particularly apparent when the dense neuron number is large. Thus, The dropout rate is usually selected according to the dense neuron number.

- **Classifier:** The classifier in our model is a fully connected layer with a softmax activation function. The number of outputs in the layer is determined by the difference between the magnification levels of the input patches x and y. For our implementation, we set n to two (Sec.4.2.2), resulting in 16 possible values for y_{true}, thus, 16 neurons in the output layer.

VGG16

VGG [10] is a convolutional neural network (CNN) model family developed by K. Simonyan and A. Zisserman from the University of Oxford. It includes multiple variants, such as VGG16 and VGG19. VGG16 is a specific variant that consists of 16 layers, including convolutional layers, pooling layers, and fully connected layers. One of the key features of the VGG16 model is its use of small 3x3 filters to extract features from input images, instead of larger filters commonly used in other models. The researchers found that using multiple smaller layers instead of a single large layer improved decision functions and allowed the network to converge quickly. Additionally, cascading 3x3 filters in VGG16 allows for more nonlinearity in the model. This approach also helps decrease the number of parameters, making the model more efficient and better at combating overfitting.

layer	channels	kernel size	stride	output size	parameters
input	3	—	—	224×224 ($224 \Rightarrow 112$)	0
2× conv	64	3×3	1	224×224 ($224 \Rightarrow 112$)	38.7
max pool	64	3×3	2	112×112 ($112 \Rightarrow 56$)	0
2× conv	128	3×3	1	112×112 ($112 \Rightarrow 56$)	221.44
max pool	128	3×3	2	56×56 ($56 \Rightarrow 28$)	0
3× conv	256	3×3	1	56×56 ($56 \Rightarrow 28$)	1475.33
max pool	256	3×3	2	28×28 ($28 \Rightarrow 14$)	0
3× conv	512	3×3	1	28×28 ($28 \Rightarrow 14$)	5899.77
max pool	512	3×3	2	14×14 ($14 \Rightarrow 7$)	0
3× conv	512	3×3	1	14×14 ($14 \Rightarrow 7$)	7079.4
max pool avg pool	512	3×3 —	2 —	7×7 ($7 \Rightarrow 3$) 1×1	0
Total (parameters are expressed in 10^3)					14714.66

Table 4.1: VGG16 backbone architecture. the red colored fields are the changes adopted in our implementation, in general, the only changes are the input dimension and the addition of global average pooling after the last max pooling layer.

In the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) of 2014, VGG16 achieved a test accuracy of 92.7%, which made it one of the top-performing models in the competition. Tables 4.1 and 4.2 show the network architecture in detail.

As shown in the tables the input images were resized to $112 \times 112 \times 3$ in order to speed up the training process by reducing the number of convolutions at each layer. While the augmentations performed include random contrast with a small factor of 0.2 for the low-resolution patch, Random rotation with the flexibility of around 320 degrees, Random flips both horizontal and vertical, big random crops of size 100×100, resizing to the original input size, and weak contrast of factor 0.1 for the high-resolution ones. In addition, the dropout was set to 0, as the fusion layer only contains a small number of neurons (256 neurons).

VGG16 classifier			Implemented classifier		
layer	output	parameters	layer	output	parameters
flatten	25088	0	Concat.	1024	0
FC	4096	102760.4	FC	256	262.14
FC	4096	16777.2	dropout	$rate = 0$	0
FC	1000	4096	FC	16	4.1

Table 4.2: VGG16 classifiers. The majority of VGG16 parameters (expressed in 10^3) come from the fully connected layers in its classifier. However, in our model, this is not a major concern as we have utilized an average pooling layer. Therefore, after concatenating both images, our combined latent vector has a total of 1024 features (512 from each image).

Light resnet

Residual networks [12] were proposed to overcome the problem of vanishing gradients in deep neural networks, which caused lower training and testing performance in very deep architectures compared to shallower ones. Although additional useless layers should ideally be rendered as identity mapping by the network, it was challenging to learn the identity function using standard convolutions.

Residual networks addressed this issue by introducing residual connections, which act as shortcuts that bypass one or more layers and feed the input directly to a later layer. Proving that such a connection allows the model to bypass unnecessary

layers and improve performance. Unlike other works such as highway networks [67, 68] that used gated connections with learnable parameters (inspired by LSTM [17]), resnet connections had no parameters when the input and output channel dimensions were the same.

Some variants of ResNet, such as ResNet-18, ResNet-50, and ResNet-101, have been developed with different depths. For our purposes, we used ResNet-18 to build our "light" version of the network by reducing channel sizes. Both the original ResNet-18 and our modified version can be found in the table...

layer	channels	kernel size	stride	output size	parameters
input	3 (3)	—	—	224×224	0 (0)
conv	64 (32)	7×7	2	112×112	9.47 (4.74)
max pool	64 (32)	3×3	2	56×56	0 (0)
4× conv	64 (32)	3×3	1	56×56	147.7 (37)
conv	128 (64)	3×3	2	28×28	73.86 (18.5)
3× conv	128 (64)	3×3	1	28×28	442.75 (110.78)
conv	256 (128)	3×3	2	14×14	295.17 (73.86)
3× conv	256 (128)	3×3	1	14×14	1770.24 (442.75)
conv	512 (256)	3×3	2	7×7	1180.16 (295.17)
3× conv	512(256)	3×3	1	7×7	7079.4 (1770.2)
avg pool	512 (256)	7×7	1	1×1	0 (0)
Total (parameters are expressed in 10^3)					11000 (2753)

Table 4.3: Resnet-18 Backbone. The residual connection happens every 2 convolutions after the max pooling layer. While a special connection that expands the channel size by linear transformation (1×1 conv) takes place when the output and input channel dimensions aren't the same. In red we have the light resnet modifications leading to one-quarter of the number of parameters.

The light resnet-18 variant has only one-quarter of the parameters of the standard resnet-18, which makes it faster to train and less susceptible to overfitting. Augmentations are applied only to the higher resolution image, including random rotation, and horizontal and vertical flipping. The network has 1024 dense neurons in the fusion layer, and a dropout rate of 0.5 is applied. As the number of channels is reduced by half for each layer, the final latent vector has half its original size. Therefore, after the concatenation layer, the network produces a vector of 512 features.

Resnet-18 classifier			Light resnet classifier		
layer	output	parameters	layer	output	parameters
FC	1000	512	Concat.	512	0
—	—	—	FC	1024	524.3
—	—	—	dropout	$rate = 0.5$	0
—	—	—	FC	16	16.4

Table 4.4: Resnet-18 classifiers.

4.3.2 Data Preparation

In this section, we will present the data creation process for our pretext task, including the steps involved in extraction and processing. Our approach to data extraction is both random and incremental, using parallelism to ensure versatility and efficiency in terms of memory and time. Each decision we make in the process is carefully considered to address resource constraints and optimize resource usage as much as possible. Through this approach, we aim to create high-quality data that is diverse, representative, and efficient for training our model

Dataset Creation

Our data creation process is designed to be both incremental and random. The incremental aspect allows us to run the creation script multiple times, producing different sets that can be combined later on. The randomness comes into play when extracting samples from a WSI, as the location is chosen randomly. This approach helps us to avoid memory and time issues that may arise if we were to extract all possible patches from a WSI. Additionally, it allows us to better represent the WSI in the dataset, as samples are extracted randomly from all over the slide, rather than following a sequence. By doing so, we can achieve a more diverse and representative dataset, with a better balance of samples from different regions of the slide.

First, we showcase the extraction process on a slide level for which We have 5 main parameters:

- **Mode:** this parameter can be either set to classification or regression, representing the formulation used. This parameter determines how the high-resolution patch is extracted and labeled. In our case, we have set it to classification.

- **Size:** the extracted patches have a width and a height equal to the value of this parameter, we set it to 256.
- **Nb:** this parameter controls the total number of samples extracted from one WSI per run or set. We set it between two to six hundred.
- **Zoom:** This parameter should be fixed when creating different subsets, in other words, it should be kept constant across the dataset when the mode is set to classification. It controls the gap n between levels x and y. In our case, we have set it to two, meaning that p_x can be one out of sixteen possible tiles.
- **Deviation:** To provide more versatility in terms of the field of view to our dataset, we allow the levels from which our patches are extracted to differ while keeping the gap parameter (zoom) constant. This is controlled by the deviation parameter, which determines the possible levels from which the high-resolution patch can be extracted. Specifically, x can be any value in the range $[0, 0 + deviation]$, and y follows it as $x + zoom$. In our case, we initially set the deviation parameter to two and degrade it for later sets.

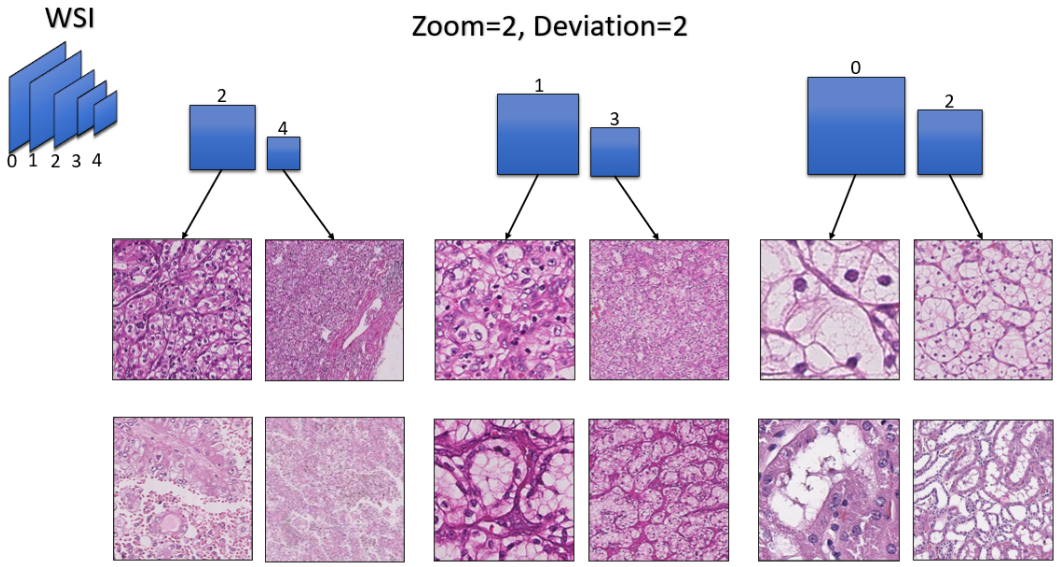


Figure 4.12: Extraction procedure. The figure above shows an example of data extraction, illustrating how samples look when deviation and zoom are set to two, and mode is set to classification. the numbers on the WSI indicate the magnification levels with 0 being the max resolution/magnification.

Each of the parameters mentioned above plays a significant role in preserving time and ensuring data correctness. Setting Nb too high can produce more samples

but at the cost of time. The zoom parameter should be carefully set to ensure task feasibility. The deviation parameter provides versatility, but it should be kept small as lower magnification levels contain fewer patches and are less informative. In fact, this is why we degrade the parameter for later sets. Figure 4.12 shows an example of the samples extracted.

Another important consideration in the image data extraction process is identifying background patches that do not contain any tissue parts. These gray patches are typically ignored and do not count toward the Nb parameter. High-resolution patches are also tested for the same condition.

Each Nb sample extracted randomly from a single slide is handled with a process out of a process pool. This means that multiple slides can be processed simultaneously, with the number of parallel processes being a parameter that is chosen depending on the machine 's capacity. Increasing the number of processes leads to a faster and more efficient extraction process.

The use of multiprocessing is particularly useful in this case since the openside library used to read and process the WSI files operates in I/O mode, where the files are too large to be loaded directly into the main memory. As a result, a lot of time is spent waiting for this operation to execute. Increasing the number of processes from 1 to 5 can improve the speed by nearly four times due to the centralized execution of the writing operation by the main process.

To optimize RAM usage, the extracted data is written to disk over time after processing a specific number of slides. This also creates checkpoints in case the process is interrupted. For example, if the process is interrupted after processing some folders, the already processed folders can be skipped, and the extraction continues from the last checkpoint that was correctly saved. This approach not only optimizes RAM usage but also provides a mechanism for resuming the extraction process in case of unexpected interruptions.

Due to randomness, running the script multiple times produces different sets with different samples. These sets can be combined into a single dataset using another script, allowing for an incremental increase in the available dataset size when needed. Initially, sets with a deviation of 2 are created, then we reset the deviation to 1 for the subsequent sets. After combining all the sets, we obtain a dataset with approximately 864,000 samples. Of these samples, 172,800 are extracted from levels $(x = 2, y = 4)$, while the remaining 691,200 samples are extracted from levels $(x = 1, y = 3)$ and $(x = 0, y = 2)$, with each level accounting for 345,600 samples.

To summarise our extraction process takes place on the training WSIs in a multiprocessing fashion including a checkpoint mechanism, fixing some parameters and varying others over time. As a result, time and memory-efficient, safe, and incremental preparation of correct, versatile, and large artificial datasets is insured by our framework.

Pre-processing Pipeline

Once the data has been extracted, it is saved into Numpy files for convenient pre-processing. Our pre-processing pipeline is designed to be resource-efficient and includes shuffling and filtering of the dataset, as well as the calculation of mean and standard deviation values. However, during our testing, we found that training with or without standardization resulted in equivalent performance. Therefore, we skip this step when training the model later on.

The final step of our pre-processing pipeline involves transforming the image files into a structure that is compatible with TensorFlow datasets. Throughout the pipeline, we process the images as matrices of unsigned integers (uint) to optimize our resources until the final reading of the data as a TensorFlow dataset. This ensures that our pipeline is efficient and effective in preparing the data for training the machine learning model. Next, we showcase the pipeline steps with their implementation:

- **Filtering:** To efficiently filter and clean the extracted data, we utilize multiprocessing, where each process filters out one Numpy file at a time and writes back the clean version. The filters used in the process make use of pixel statistics and compare them to specific thresholds to determine whether or not to remove a sample. One filter calculates the ratio of white or gray pixels to the size of the image, providing an estimate of how much of the image is background. Another filter attempts to identify color standard deviation between pixels, as some patches may have a single color region outside of the tissue due to dirty glass or other factors. The thresholds used for these filters were chosen manually to ensure optimal cleaning of the dataset. We set the thresholds for color's standard deviation filters at 4.5 and around 50% for white (gray) pixel filters. After applying these filters, our dataset was cleaned and resulted in 784,495 samples.

- **Shuffling:** Due to the size and structure of our dataset, all the extracted files belong to the same cancer type. To avoid memory overload, we perform partial shuffling of the dataset using multiprocessing. Each process takes two files at a time, concatenates them, shuffles the concatenated dataset, and splits it into two equal parts. The shuffled parts are then written back to memory.

Since the shuffling procedure is limited to two files, we run it multiple times to ensure that the dataset is well-mixed.

- **Splitting:** To train our model effectively on our self-supervised task, we require a metric to detect any overfitting issues and to select the best hyperparameters. Therefore, we must create a validation dataset from our large collection of samples. For a massive dataset such as ours, setting aside only 8% of the data for validation and using the remaining 92% for training is sufficient. In addition, we ensure that our validation set is representative of the training one by employing a splitting strategy that splits the data file by file. As a result, our split comprises 721,737 training samples and 62,758 validation samples.
- **Tensorflow Dataset:** Once the dataset is shuffled, filtered, split, and structured as Numpy files of appropriate sizes, we utilize the generator function of the TensorFlow dataset API to read the data. With this API, we normalize the pixel values to a range of 0 to 1, transform them into float32, and perform a local shuffling of the samples. To expedite the training process, we also employ the prefetch function. By utilizing these techniques, we optimize RAM usage while maintaining a reasonable training time.

Finally, the obtained dataset contains samples with the following structure $[[p_x, p_y], y_{true}]$ with $[p_x, p_y]$ being an array with shape (2,256,256,3) which are further rescaled 224 for Light-resnet and 112 for VGG16.

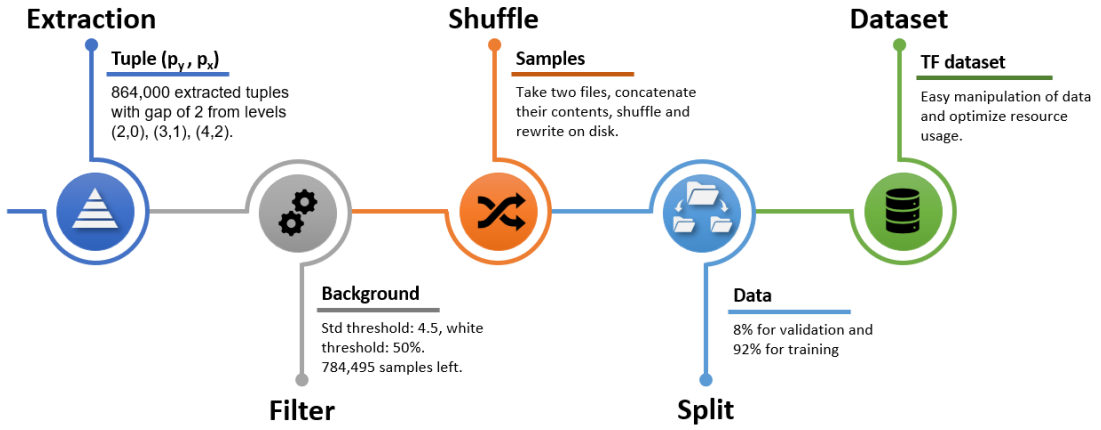


Figure 4.13: SSL data pre-processing pipeline.

4.3.3 Training

Here we discuss the main challenges we encountered while training our model and the solutions we employed to overcome them. Additionally, we present some

potential solutions that we did not implement but could be useful in addressing these challenges. This section is divided into two subsections, one that focuses on Light-resnet training, and the other addresses VGG16 training.

Light-resnet training

At the beginning of our experiments, we prepared a set of 20,000 training samples to validate our task. The first model we tested was the standard ResNet-18, which was able to reach 100% training accuracy. However, when we looked at the validation accuracy, we noticed that it remained stagnant at 6%. We attempted to change hyperparameters, but the best result we could achieve at this stage was only 12% validation accuracy, indicating severe overfitting. Transfer learning did not improve performance, likely because our task was significantly different from natural image classification tasks. Both our problem domain and task nature are different, in addition, our task even differs from localization tasks.

The main contrast between object localization and our patch localization lies in the nature of the features required for each task. Object localization involves identifying and extracting specific repeated patterns to enable recognition and detection of the object. On the other hand, patches are not objects they are pieces from an object, a background, or both combined. to locate them, one should learn a general set of repeated features from high and low-resolution patches to understand the probable region that contains the high-resolution patch using fusion. As a result, the approach required to solve patch localization tasks is markedly different from that used for standard detection tasks.

The only solution was to increase the training dataset size. Fortunately, due to the incremental nature of our data extraction framework, this was relatively easy to accomplish. We added an extra 100,000 samples to the training set and ran some experiments using different hyperparameters. This resulted in a remarkable improvement over our previous results, as shown in Figure 4.14.

However, despite this improvement, we still encountered overfitting issues relatively early in the training process. To overcome this, we decided to increase the size of our dataset even further, until it reached a total of 480,000 samples. We also opted to use the light-resnet architecture for training, as it is both easier and faster to train while having fewer parameters. These data were generated using the initial split (section 3.2), in addition, light-resnet doesn't have an ImageNet-initialized weights [9]. Thus, we decide to use it only for insights and not for our final studies and evaluations.

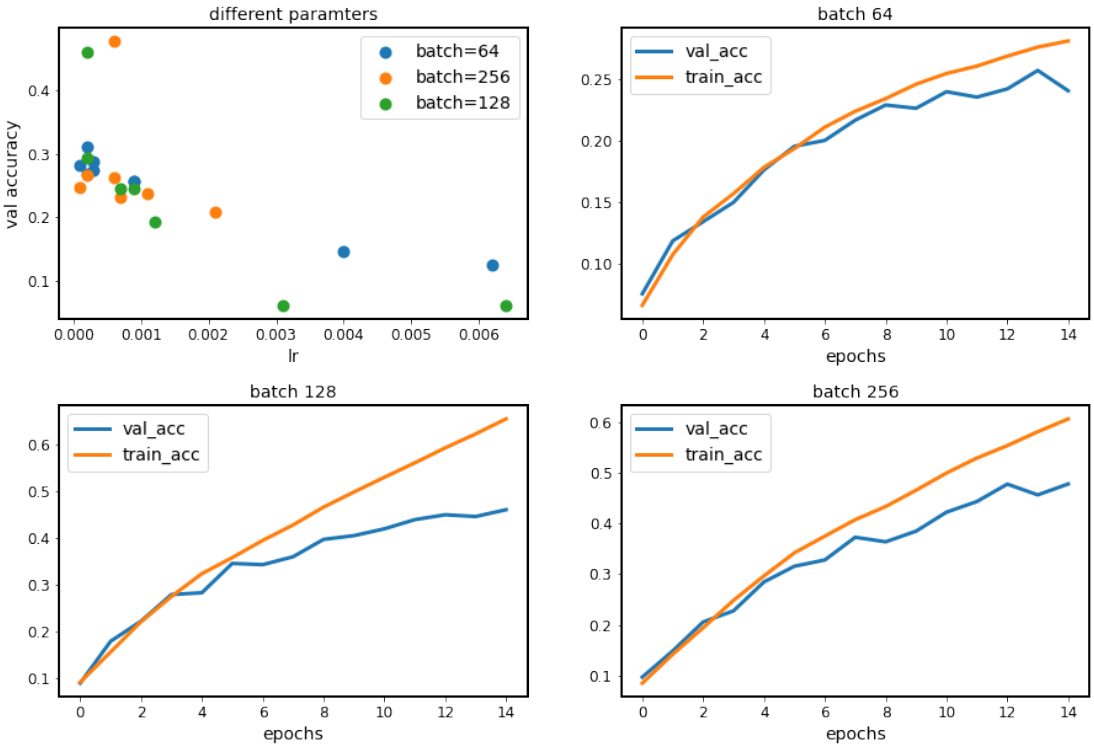


Figure 4.14: Resnet-18 training insights. Trying different learning rates and batch sizes on the 120,000 samples training dataset. Note that the best learning rates obtained for each batch size in the scatter plot are used in the other graphs.

With the large dataset, The trend of increasing the learning rate for an increased performance didn't hold as in 4.14 due to augmentation, dropout, and the new samples. We rerun the hyper-parameters tuning for light-resnet obtaining the following parameters: lr= 0.0003, weight decay= 0.00001, batch= 256, using Adam as optimizer with its default beta1 and beta2. We train the model for 200 epochs while saving the weights with the best validation accuracy and stop the training when performance stops improving for a certain amount of epochs. Figure 4.15 shows light-resnet training. The highest performance we reached was around 70% validation accuracy before it degraded.

It's important to note that all of the data used here are not part of our final study as they were built upon the initial patient split (limited test split). Thus, they were only explained here to show how severe overfitting is in our task and to motivate the huge dataset size we built and described in subsection 4.3.2.

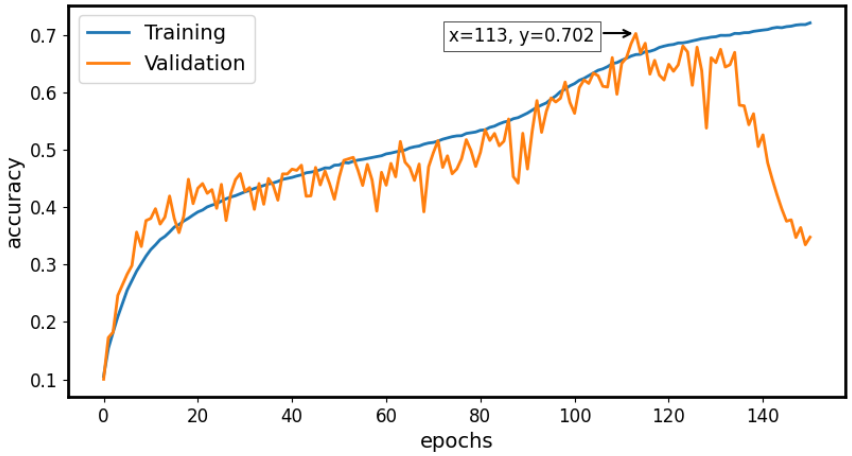


Figure 4.15: Light-resnet training.

VGG16 training

Training VGG16 on our self-supervised task posed a number of challenges due to the high number of parameters, large dataset, and complex nature of the task. In this section, we will detail these challenges and describe the decisions we made to overcome them.

- **Overfitting:** As previously mentioned, we encountered severe overfitting issues when training with the light-resnet model. To address this problem, we decided to build a much larger dataset (subsection 4.3.2) and add additional augmentation layers such as crops and contrast.
- **Slow training:** To address the challenge of long training time due to the complexity of the VGG16 network and the large dataset, we made two key modifications. First, we rescaled the input image size from 224×224 to 112×112. This helped minimize the number of convolution operations required during training. Secondly, we set the dropout rate to 0. As a result of these modifications, we were able to train one epoch in approximately 15 minutes using an NVIDIA A100 GPU.
- **Hyper-parameters tuning:** To mitigate the challenge of long training time and improve our search for optimal hyperparameters, we made several adjustments to our approach. Rather than running a standard hyperparameter tuning process on the entire dataset, we limited this step to using only a fraction of the data. Additionally, we manually selected a set of hyperparameters to change and carefully studied their effects to make informed decisions.

- **Learning-rate sensitivity:** We observed that the training process is highly sensitive to the value of the learning rate. In fact, there exists a small range of learning rate values, typically between 5×10^{-5} and 10^{-5}, where learning is feasible and effective. Additionally, due to the low initial value of the learning rate, the network is at risk of getting stuck in bad local optima. To mitigate this risk, we implemented a manual warm-up [69] process by increasing the learning rate after some epochs. This approach helped to improve the stability of the training process and avoid getting trapped in sub-optimal solutions.

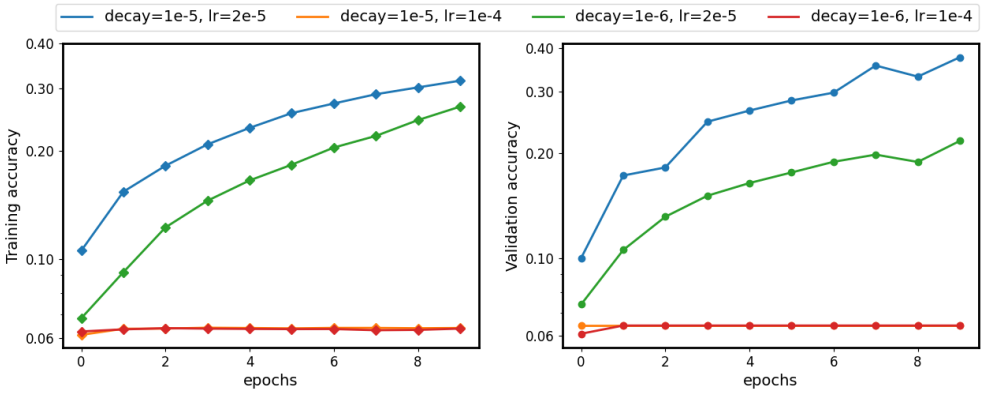
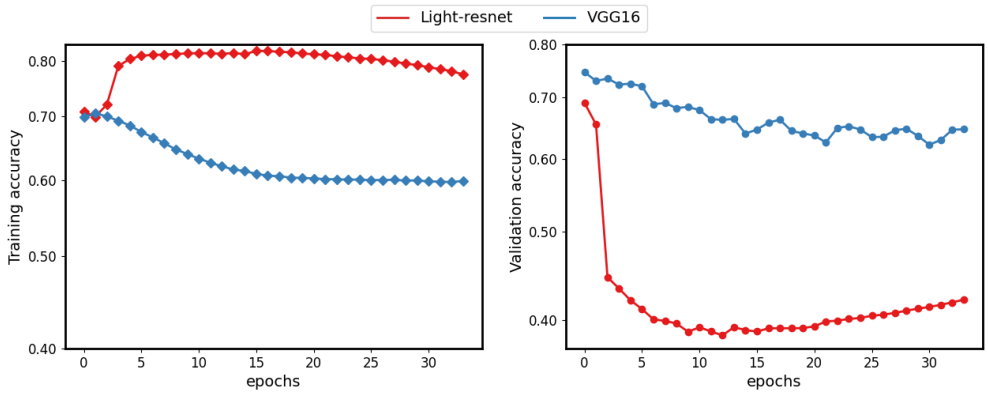


Figure 4.16: Learning rate sensitivity.

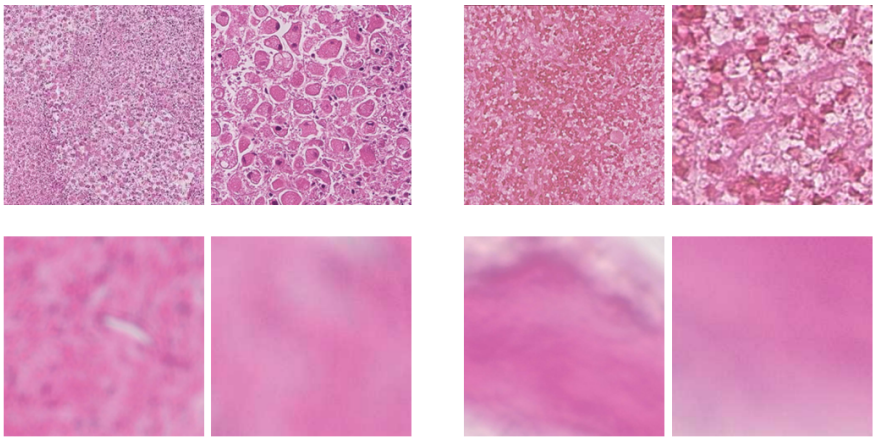
- **Decaying the learning rate:** In our training process, we encountered an anomaly when decreasing the learning rate to break through a plateau. While this method usually leads to better performance, we observed that our model's performance on the training set initially increased, but then gradually degraded (Fig. 4.17a). We believe that this was caused by overfitting complex samples (Fig. 4.17b) in our dataset, which introduced noise instead of useful knowledge. However, the initial increase may have been due to better learning of normal patterns. To address this issue, we considered two solutions: reducing the weight decay when decreasing the learning rate to allow the model to fit the data better while still having some constraints, or increasing the batch size [70] instead of decreasing the learning rate. However, both approaches can also increase overfitting. This was a challenging problem, as it is difficult to automatically determine which samples are informative during dataset generation and other factors may also be contributing to the issue.

- **Forgetfulness:** While this is a major issue when the model is switched to train on a new task [71, 72], it also slightly exists when training the same task on a large dataset [73], as the model tends to forget some of what it learned on earlier batches when continuing with the training process. However, using

an appropriate batch size can help mitigate this issue.



(a)



(b)

Figure 4.17: Decaying learning rate effect. In (a)m the learning rate is divided by 10 in both light-Resnet and VGG16 after reaching a plateau. In (b), we show some examples of hard and noisy inputs.

Aside from the solutions we have already implemented, there are other techniques that could be utilized to further enhance and streamline the training process. For example, cosine annealing [74], knowledge distillation [75], and curriculum learning [76, 77, 78] are all potential methods that could be employed. Curriculum learning, in particular, could be highly effective for our task as it involves gradually increasing the difficulty of the training data over time which can be easily implemented in our pretext task. By implementing this approach, we could further improve the training process and potentially achieve even better results.

We Train VGG16 with an initial 2×10^{-5} learning rate, a batch size of 32, and a 10^{-5} weight decay, using Adam as optimizer with beta2 set to 0.999 and beta1 left unchanged. We vary this parameter later on upon reaching a plateau multiple times except for the optimizer exclusive parameters (beta1 and beta2).

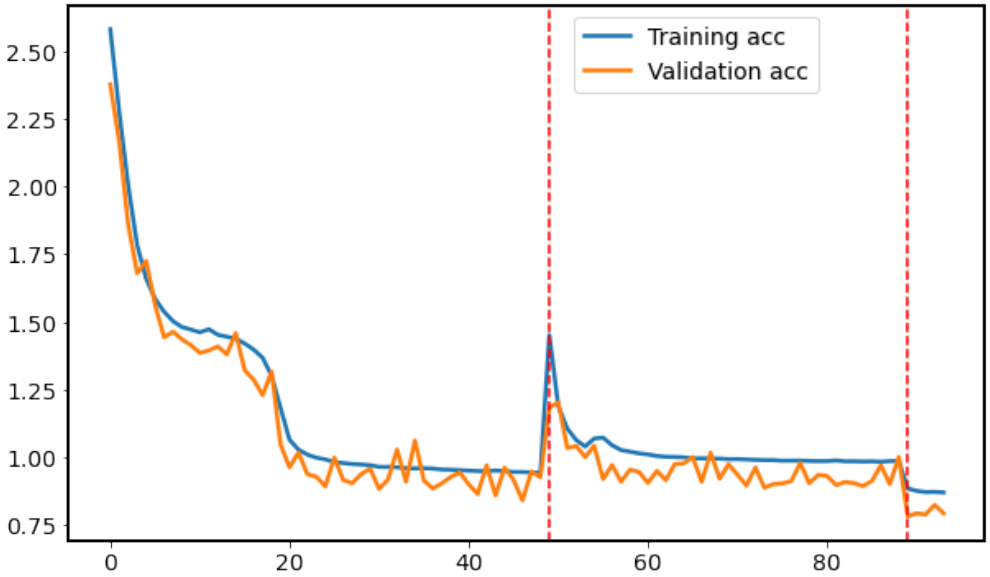


Figure 4.18: In VGG16 training, the red lines signify that some parameters have been changed. The first change is increasing the learning rate to 10^{-4}, while the second change is increasing the batch size to 64.

4.4 Observations

In order to gain insights into what our model has learned and its capabilities, we utilized interpretability methods designed for CV tasks such as Grad-CAM [79]. We provide a brief overview of Grad-CAM, its workings, and how we implemented it for our model. Finally, we conclude this chapter with some interesting findings that we observed.

Grad-CAM (Gradient-weighted Class Activation Mapping) is a popular interpretability technique used in CNN. It is an extension of the earlier technique known as CAM [80] (Class Activation Mapping). CAM requires the network to be fully convolutional with only one fully connected layer acting as a classifier. It computes the importance of different activation maps outputted by the last convolutional layer by using the weights connected to the predicted class. Finally, it performs a weighted sum of these maps to form a heatmap of the image with the most relevant

regions affecting the output having the highest scores.

Grad-CAM follows a similar process, but it allows the model to have different types of layers beyond just the convolutional ones. Rather than directly using the weights, Grad-CAM uses the gradients to determine the importance of each feature map. These gradients are backpropagated through the network until they reach the last convolutional layer, where they are used as the importance weights for the different activation maps. This approach can be used with a wider range of neural network architectures. However, when the network has only one fully connected layer, both CAM and Grad-CAM become equivalent.

Our network incorporates multiple fully connected layers to support a fusion layer, thus, we built the heatmaps using grad-CAM. In our implementation, the gradient is split into two after reaching the fusion layer, which produces a different weight array for each input image (p_x and p_y). Finally, we rescale the heatmap to match the input image size of 256×256 for better visualization.

We can observe that the process of generating heatmaps is closely related to the size of the activation maps, which, in turn, depends on the size of the input images. The Light-resnet architecture produces activation maps of size 7×7, while the VGG16 architecture produces maps of size 3×3. Therefore, by utilizing Light-resnet here, we can generate higher-resolution heatmaps that provide better highlighting of the regions that significantly influence the model's output.

In Figure 4.19, we observe that the model's attention is not always focused on the zoomed-in patch region. Instead, it often scans the entire zoomed-out patch which helps it exclude some patches. This behavior is akin to a student omitting certain options in a multiple-choice exam. This pattern is evident across multiple inference instances and is sometimes accompanied by the typical behavior seen in image classification tasks. This observation is particularly noteworthy, as it suggests that the model has learned an exclusion rule, which is not typically observed in image classification models that tend to focus on the entire or specific parts of the object being classified.

To further validate our observation, we examined the weights between the fusion layer and the output layer. We calculated some statistics and discovered that the minimum weight had a larger absolute value than the maximum weight by multiple times. Typically, in a network with the RELU activation function, active neurons have positive values. Thus, negative weights play a role in deactivating their corresponding output neurons. However, this behavior is not typical in image classification networks. We showcase these statistics and compare them to the

weights of a resnet50 trained on imagenet in table 4.5. For all networks, we show $top_5(max(W))$ where max is calculated with respect to the output neuron.

model		1	2	3	4	5
Resnet50	max	0.74	0.7	0.68	0.67	0.67
	min	-0.21	-0.2	-0.18	-0.17	-0.17
Light-resnet	max	0.077	0.068	0.067	0,065	0.065
	min	-4.68	-4.67	-4.53	-4,5	-4.5
VGG16	max	0.0073	0.0063	0.0063	0.006	0.0051
	min	-0.375	-0.37	-0.368	-0.366	-0.364

Table 4.5: Weights, top5 maximum and minimum classifier weights of three different models: Imagnet trained Resnet50 and both our self-supervised trained networks

This chapter presented the methodology we developed for our research, motivating it by a comparison with existing methods in the field. We provided a detailed explanation of our formulation, implementation, and the challenges we encountered along the way, including the solutions we adopted to overcome them. In addition, we showcased the interesting patterns that our model learned using Grad-CAM and further validated our approach by examining the classifier weights after opening up the network.

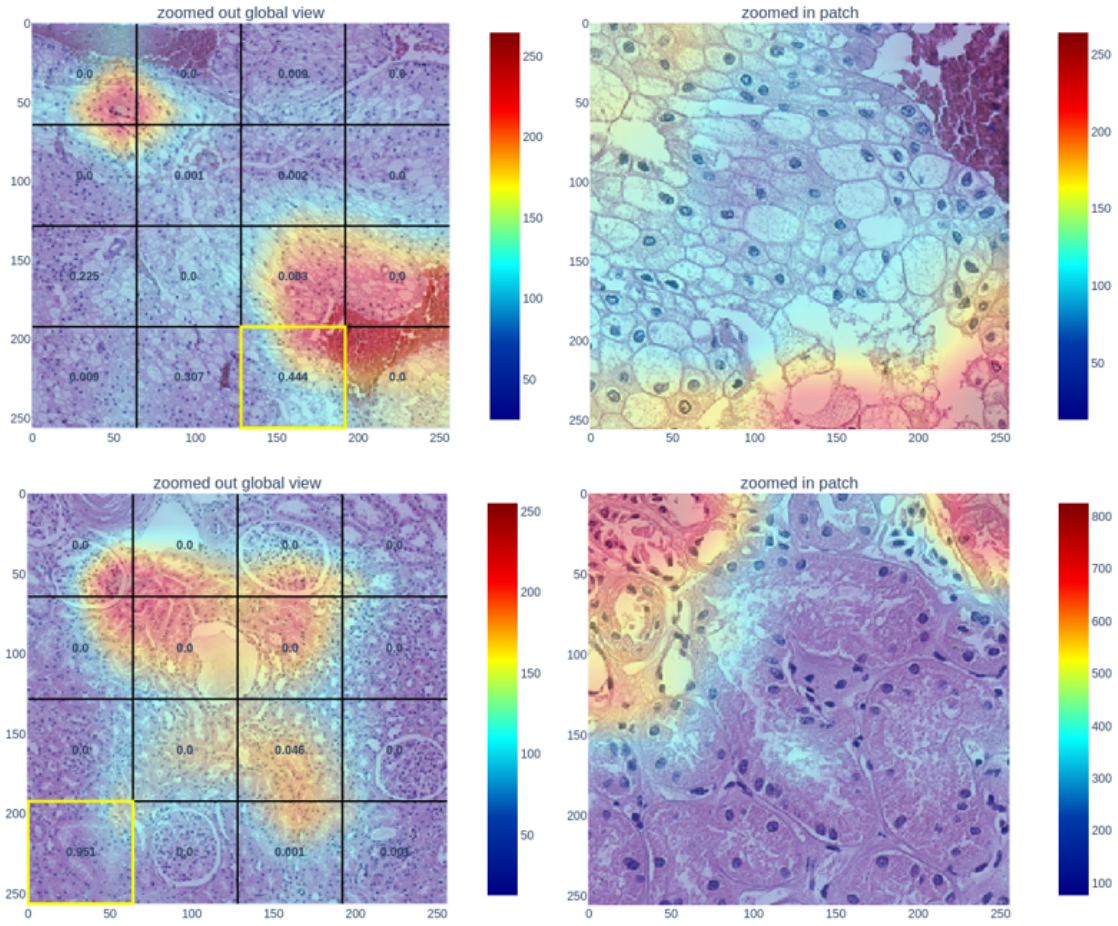


Figure 4.19: Heatmaps.

Chapter 5

Results

In this chapter, we will walk through the data preparation process for our Main task of RCC subtyping. Additionally, we will present an experimental section where we compare our pretext-task approach to both transfer learning and the method proposed by Ding et al. [61].

5.1 Data Pre-processing

To begin with, we present our dataset annotation format, which is a crucial component of the pre-processing pipeline. In the case of chromophobe and oncocytoma, there exist annotation folders that include crops of the tumor regions extracted from the WSI in the respective data folders. On the other hand, for ccRCC and pRCC, XML files in ASAP format are used containing ROI annotations of both tumorous and non-tumorous regions including fiber, necrosis, and normal cells. Figure 5.1 shows an overall view of the pipeline.

```
<ASAP_Annotations>
<Annotations>
<Annotation Name=... Type=... PartOfGroup="necrosis" Color=...>
    <Coordinates>
        <Coordinate Order="0" X="60130.6914" Y="152329.781"/>
            ....
        <Coordinate Order="44" X="60345.6523" Y="152701.078"/>
    </Coordinates>
</Annotation>
    ...
</Annotations>
</ASAP_Annotations>

```

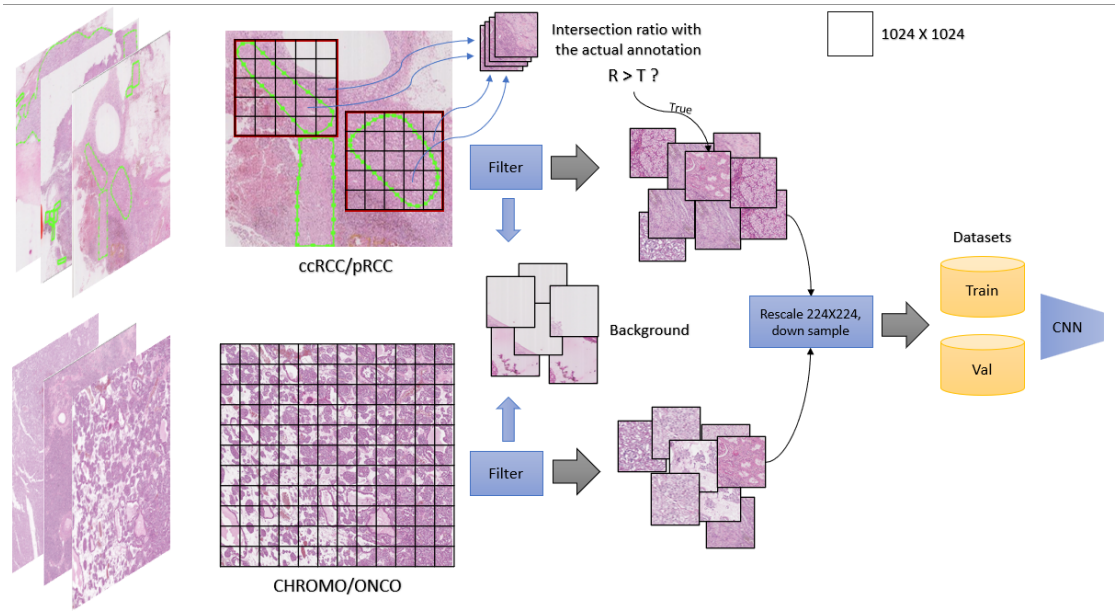


Figure 5.1: Main task pre-processing pipeline.

Starting with XML annotation files, Each ROI is included in a `<Annotation>` field, which contains a `PartOfGroup` element specifying its type. The coordinates are represented as subfields that define the boundaries of the annotation. To begin the patch extraction process, we first extract all annotations and represent them as a list of points along with an element that describes the type of tissue contained within each ROI. Next, we define a box that encloses the ROI, as shown in Figure 5.1, and divides this box into a set of non-overlapping tiles measuring 1024×1024 at the highest resolution level. Finally, we obtain a list of square patches for each annotation, which can be used for further analysis or processing.

We calculate the intersection area over square area ratio for each patch by determining the area of intersection between the patch and the actual annotation. When the square patch lies completely within the actual ROI, this ratio is equal to one, whereas when the patch is completely outside the ROI, the ratio is equal to zero. Next, we define a threshold to filter out patches with a low intersection. Since the area around the actual ROI is mostly homogenous with what is inside it, we set a smooth threshold of 10-20%. This means that patches with a higher ratio than the threshold are considered while those with a lower ratio are ignored.

For chromophobe and oncocytoma samples, we split the tumor crops directly into a grid of non-overlapping tiles. However, there is one key difference, as the

edge tiles may overlap with the previous ones if they are not large enough to reach the size of 1024×1024. After The extraction phase, we end up with a list of tuples containing a patch and its label.

We apply the same filtering procedure described in Section 4.3.2 to remove background elements from the list of patches. We then perform a primary downsampling step to balance the dataset and remove excess patches. This step involves extracting the minority class data first and determining the number of samples available for them. We then extract a metadata file that contains the number of annotations for each class by WSI, which we use to gain insights into the average number of class annotations present within a file, as well as other statistics such as the total number of files per class.

By combining these statistics with the number of samples available for the minority class, we advise an upper limit on the number of patches that can be extracted from a single annotation. Since this is only a primary step, we use a soft limit that allows for the total number of samples for the majority classes to be around five times the number of available samples for the minority class. The sampling of patches from a single annotation is not uniform and is based on a probability distribution proportional to the intersection over square area ratios.

The list of patches extracted from a single WSI is saved and the whole process is performed in parallel by multiple processes. When this step ends, we read the whole dataset and rescale all the patches to 224×224, perform the final downsampling, and split the data into validation and train sets (85% and 15% of data respectively for each set).

Dividing the process into two phases allows for more flexibility in terms of how the data is processed for downstream tasks. For instance, instead of downsampling directly to 224×224, we could crop the patches into smaller ones representing different magnification views. Thanks to the initial step of extracting the patches with the size of 1024×1024, we do not need to rerun the full pipeline to perform different processing.

Finally, for the test set, we perform data extraction in two different ways. One approach addresses patient-level tests, and the other approach tackles patch-level tests. For the patch-level test, the extraction procedure is aligned with the training data extraction; however, no undersampling is needed. For the patient-level test, we extract all patches that are not background from the WSI, regardless of whether they belong to an ROI or not. Then, we assign the label to the patient based on a majority voting method using the patches from that patient.

5.2 Experiments

We conducted a comparative analysis of our proposed method against two other approaches, transfer learning and *Ding et al.* [61] method, which is the closest to our method regarding the fundamental idea. We chose to compare our method only with *Ding et al.* due to the high cost of extracting data and training different pretext tasks, and because they had already tested their method against various other ones in the literature. To evaluate the effectiveness of our method, we conducted empirical experiments on patient and patch-level for our dataset.

To train Vgg16 using *Ding et al.* pretext task, we started from our self-supervised task dataset and generated data using a simple transformation. We took a tuple of samples and decided to switch their high-resolution patch with a probability of 0.5. If the patches were switched, both samples became negative samples, and if they were not switched, the samples were considered positive. We trained the network for around 50 epochs using the hyperparameters utilized by Ding et al.

Training

We Transfer the self-supervised trained networks to the downstream task of cancer subtyping where both networks are trained in two stages. The first one act as an initialization of the classifier weight where the models are trained for 4 epochs while freezing their backbones. A learning rate of 10^{-3} is used for the first 2 epochs and it is divided by 10 for the next 2. In the second stage, we unfreeze the backbones and deploy a cosine warmup, which allows the learning rate to reach the maximum value of 10^{-4} within 5% of the total training iterations. We train the models for 120 epochs using a batch size of 2. The best-performing models in terms of validation accuracy are saved.

For each self-supervision task, we tested two network weight sets: the first was the base set after pre-training the network as discussed earlier in the self-supervised task, and the second was a fine-tuned version on the self-supervised task, which we called "finetuned". The finetuned version was simply a fine-tuning of the base version on the pretext task with a lower learning rate.

We experiment with two versions of ImageNet pre-trained VGG16 models. The first version is trained with a batch size of 2 and a learning rate of 10^{-4}. For the second version, we freeze the entire network except for the last convolution layer and train it with a batch size of 128 and a learning rate of 10^{-5}. Both models are trained for 120 epochs, the best-performing models are also saved similarly to the self-supervised networks. All the mentioned models are trained using Adam

optimizer with the default set of parameters and a weight decay of 10^{-5}.

Discussion

Our study demonstrates that our model outperforms all the other models at the patient level (Fig. 5.2), with both the finetuned and base versions performing well. The finetuned version prioritized Onco over Chromo, achieving 3 out of 4 correct predictions for Onco and 1 out of 3 for Chromo. Meanwhile, the base version was fairer towards Chromo, achieving 2 out of 4 correct predictions for Onco and 2 out of 3 for Chromo. Both versions shared one Chromo patient wrongly predicted as ccRCC and one Onco patient wrongly predicted as pRCC. Considering pRCC, both versions correctly predicted all 7 patients. However, for ccRCC, the base version outperformed the finetuned version with one patient, as the latter had one additionally misclassified patient as pRCC.

On the other hand, the finetuned version of Ding et al.'s initialized network outperformed the base one, especially with respect to the ccRCC class, with only one patient difference. Both versions had 2 correctly classified Onco, 1 correctly classified Chromo, and all 7 pRCC correctly classified. Once again, the additionally misclassified ccRCC in the base version was wrongly assigned to pRCC. The finetuned version had 7 wrong predictions, compared to 8 wrong predictions for the base one, while our base version had only 5 wrong predictions.

Finally, the transfer learning trained networks ranked last, with the unfrozen version performing poorly, with 11 misclassifications. The highest misclassification rate was for ccRCC, where 7 patients were assigned to pRCC, 1 pRCC patient was assigned to Onco, and 1 Onco patient was wrongly assigned to pRCC. Chromo had only 1 correct prediction. However, the frozen version performed better with 9 misclassifications, most of them coming from ccRCC. This time, only 3 were misclassified as pRCC while the rest of the mispredictions accounting for 3 were assigned to Onco. This version is considered the best, in terms of Onco and Chromo predictions, achieving 3 out of 4 correct Onco predictions and 2 out of 3 accurate Chromo predictions. Lastly, it had only one pRCC patient misclassified as ccRCC.

In our patch-level analysis, we present the per-class recalls and macro F1 scores for each model. We chose these metrics due to the unbalanced nature of our dataset. Although balanced accuracy is another option, it only considers recall, whereas the F1 score takes both recall and precision into account. AUC score is another alternative, which we will mention in the following section, along with the model that achieved the highest score. Overall, we present our results in terms of the F1 score as our primary metric.

Table 5.1 clearly demonstrates that our method outperforms both Ding et al. and transfer learning, achieving an impressive F1 score of 0.865. What's more, our model achieves the highest balanced accuracy and AUC of 0.876 and 0.983, respectively. When it comes to per-class recalls, our technique excels in identifying chromo, onco, and ccRCC, achieving the highest recall scores for these classes (0.987, 0.873, and 0.873, respectively). Notably, the no-freeze version of the transfer learning model achieved the second-highest recall score for onco, with a score of 0.76, but this still represents a 10% gap from our fine-tuned technique.

Ding et al.'s method achieved the highest per-class recall for non-tumor and pRCC with values of 0.946 and 0.877, respectively. Although this method performs well for pRCC and ccRCC, it performs poorly for onco, having the lowest recall of approximately 0.53.

In addition, it is worth noting that the frozen version of Transfer learning attained the lowest F1 score of 0.76, while the unfrozen version achieved a score of 0.835, ranking it second. This is contrary to the results obtained for each version at the patient level where the frozen version outperformed the unfrozen one.

In summary, our model achieves impressive results on both patch and patient levels, outperforming both the Ding et al. method and transfer learning. Our model exhibits the highest F1 score, balanced accuracy, and AUC score, with a per-class recall that excels in identifying chromo, onco, and ccRCC. Additionally, our model achieved the lowest cross-entropy loss, indicating high confidence in its classifications. Overall, our results demonstrate the effectiveness of our method in accurately classifying renal cell carcinoma subtypes.

Table 5.1: Patch level results.

models		Non-Tumor	ccRCC	pRCC	Chromo	Onco	F1 score
TL	Freeze	0.873	0.755	0.866	0.86	0.66	0.76
	No freeze	0.94	0.815	0.851	0.85	0.767	0.835
[61]	Finetuned	0.927	0.856	**0.877**	0.945	0.532	0.817
	Base	**0.946**	0.821	0.87	0.846	0.535	0.795
Ours	Finetuned	0.929	0.846	0.82	0.912	**0.873**	**0.865**
	Base	0.934	**0.873**	0.767	**0.987**	0.666	0.835

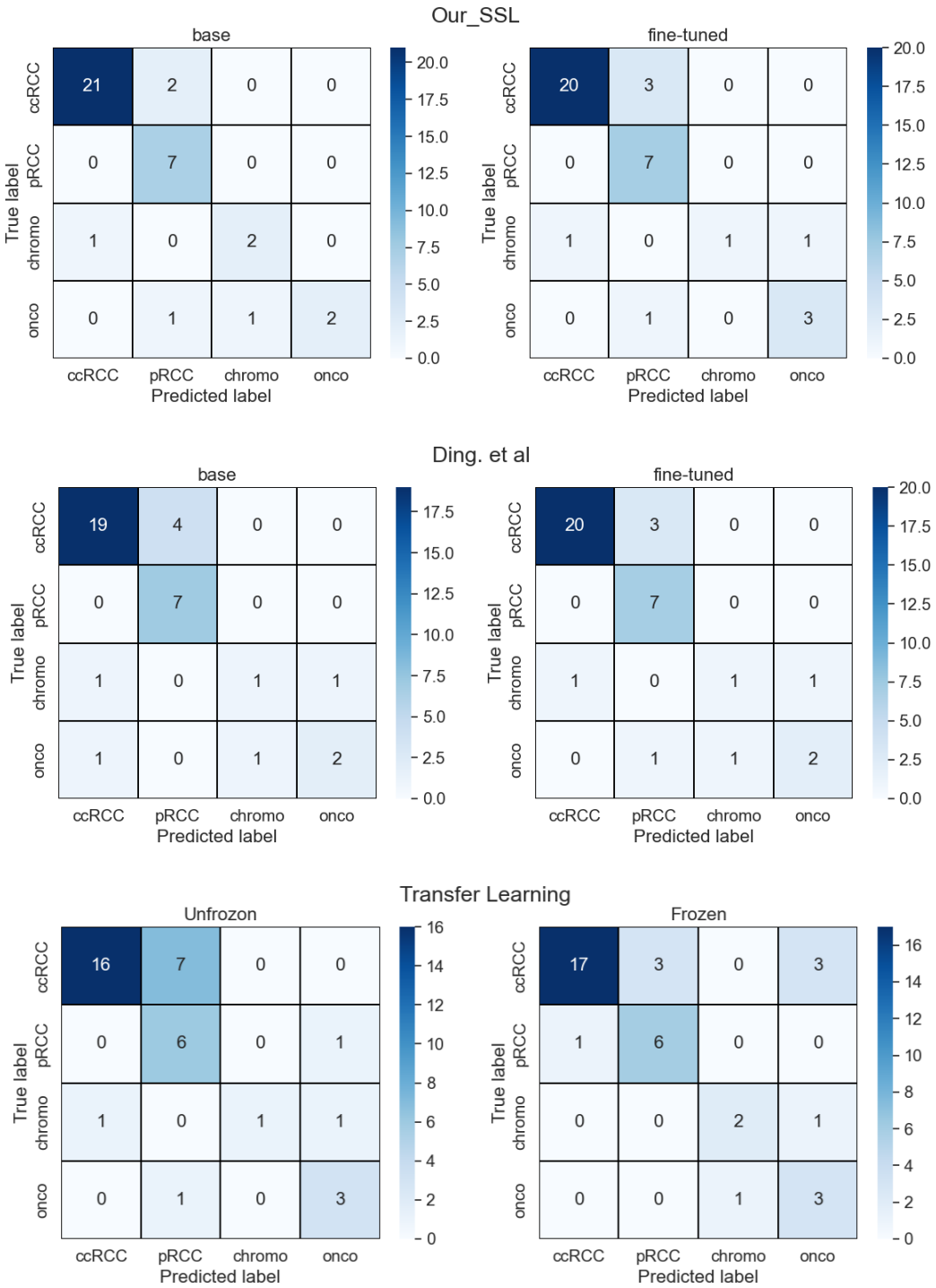


Figure 5.2: Patient level results

Chapter 6

Conclusion and Future Work

Our project aims to address the critical task of subtyping Renal cell carcinoma, which is a significant and prevalent cancer worldwide. Specifically, we focus on the four major subtypes: ccRCC, pRCC, Chromophore, and oncocytoma. Accurate categorization of these subtypes is crucial for effective prognosis and treatment. The task is approached using deep learning techniques, and we highlight a major challenge posed by the scarcity of training data due to the high cost of annotating whole slide images and the rarity of some subtypes.

To overcome this issue, we propose a novel pretext task based on self-supervised learning that leverages the interconnection between high and low-resolution patches. Our method involves a localization-driven task where the model is presented with low and high-resolution patches, with the high-resolution patch being located inside the lower-resolution one. The model must use features from both resolutions to accurately localize the magnified patch within the global view.

We constructed a large-scale artificial dataset, carefully documenting each step and rationale behind our decisions. We utilized this dataset to train a Vgg16 network on our proposed pretext task and transferred the learned weights to compare against the weights of a Vgg16 network trained using Ding et al.'s method and an Imagenet initialized network. Our evaluation focused on the downstream task of RCC subtyping, where we demonstrate that our self-supervised learning approach outperforms transfer learning and achieves similar or better results than Ding et al.'s method, and in some cases, surpasses it.

Additionally, we provide insights into the training process and decision-making of the networks trained on our pretext task, highlighting possible challenges and strategies to overcome them. While we only implemented a straightforward and easy version of our pretext task, we presented two possible formulations, including

a regression-based formulation and a classification-based one. In our future work, we plan to investigate the regression-based formulation, which aligns more naturally with localization-based objectives.

Another noteworthy aspect is the nature of our pretext task, it does not only involves capturing the relationship between high and low magnifications but is also more effectively solved by zooming in on specific regions in the low-resolution patch. Hence we introduce another interesting part of our future work.

"pathologist zooms in and out into each region, where the tissue is examined at high to low resolution to obtain the details of individual cells and their surroundings" [57]. To emulate this process, an ideal framework would be able to dynamically zoom in and out of WSIs, capturing a sequence of images that can be processed to make a final decision about the cancer subtype present in the WSI. This framework would enable a more comprehensive and accurate analysis of the tissue samples.

A complex framework utilizing Reinforcement Learning (RL) can be employed to implement this approach. The agent's actions can include zooming in or out, as well as moving up, down, left, and right, culminating in a final decision output which represents the 4-class classification in our case. This framework works on WSI level meaning that fewer training samples are available while the model complexity is higher. However, in the best-case scenario, a working model following this methodology is efficient, scalable, and better interpretable.

On the other hand, the nature of our pretext task accommodates well such a framework where the network's decision becomes the location of the high-magnification patch. This allows for even more challenging samples with larger zoom gaps, unlocking the full potential of our proposed self-supervised pretext task. Exploring the potential of RL for cancer subtyping and developing more sophisticated RL algorithms tailored to this task could be a breakthrough in WSI analysis and cancer subtyping making it a key element of our future work.

Bibliography

[1] James Hsieh, Mark Purdue, Sabina Signoretti, Charles Swanton, Laurence Albiges, Manuela Schmidinger, Daniel Heng, James Larkin, and Vincenzo Ficarra. «Renal cell carcinoma». In: *Nature Reviews Disease Primers* 3 (Mar. 2017), p. 17009. DOI: 10.1038/nrdp.2017.9 (cit. on p. 1).

[2] Ghislaine Scelo and Tricia L. Larose. «Epidemiology and Risk Factors for Kidney Cancer». In: *Journal of Clinical Oncology* 36.36 (2018), pp. 3574–3581. DOI: 10.1200/JCO.2018.79.1905 (cit. on p. 1).

[3] Holger Moch, Antonio Cubilla, Peter Humphrey, Victor Reuter, and Thomas Ulbright. «The 2016 WHO Classification of Tumours of the Urinary System and Male Genital Organs-Part A: Renal, Penile, and Testicular Tumours». In: *European urology* 70 (Feb. 2016). DOI: 10.1016/j.eururo.2016.02.029 (cit. on p. 1).

[4] Sairam Tabibu, P K Vinod, and C. Jawahar. «Pan-Renal Cell Carcinoma classification and survival prediction from histopathology images using deep learning». In: *Scientific Reports* 9 (July 2019). DOI: 10.1038/s41598-019-46718-3 (cit. on pp. 1, 9).

[5] B Escudier, C Porta, M Schmidinger, N Rioux-Leclercq, A Bex, V Khoo, V Grünwald, S Gillessen, A Horwich, and ESMO Guidelines Committee. Electronic address: clinicalguidelines@esmo.org. «Renal cell carcinoma: ESMO Clinical Practice Guidelines for diagnosis, treatment and follow-up». en. In: *Ann. Oncol.* 30.5 (May 2019), pp. 706–720 (cit. on p. 2).

[6] Umberto Capitanio, Karim Bensalah, Axel Bex, Stephen A Boorjian, Freddie Bray, Jonathan Coleman, John L Gore, Maxine Sun, Christopher Wood, and Paul Russo. «Epidemiology of Renal Cell Carcinoma». In: *European urology* 75.1 (Jan. 2019), pp. 74–84. ISSN: 0302-2838. DOI: 10.1016/j.eururo.2018.08.036. URL: https://europepmc.org/articles/PMC8397918 (cit. on p. 2).

[7] Jacques Ferlay, Isabelle Soerjomataram, Rajesh Dikshit, Sultan Eser, Colin Mathers, Marise Rebelo, Donald Maxwell Parkin, David Forman, and Freddie Bray. «Cancer incidence and mortality worldwide: sources, methods and major patterns in GLOBOCAN 2012». en. In: *Int. J. Cancer* 136.5 (Mar. 2015), E359–86 (cit. on p. 2).

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. «ImageNet Classification with Deep Convolutional Neural Networks». In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger. Vol. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf (cit. on p. 3).

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. «ImageNet: A large-scale hierarchical image database». In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848 (cit. on pp. 3, 4, 44).

[10] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2014. DOI: 10.48550/ARXIV.1409.1556. URL: https://arxiv.org/abs/1409.1556 (cit. on pp. 3, 9, 35, 36).

[11] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. *Going Deeper with Convolutions*. 2014. DOI: 10.48550/ARXIV.1409.4842. URL: https://arxiv.org/abs/1409.4842 (cit. on p. 3).

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *Deep Residual Learning for Image Recognition*. 2015. DOI: 10.48550/ARXIV.1512.03385. URL: https://arxiv.org/abs/1512.03385 (cit. on pp. 3, 9, 35, 37).

[13] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015. DOI: 10.48550/ARXIV.1502.03167. URL: https://arxiv.org/abs/1502.03167 (cit. on p. 3).

[14] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. *Instance Normalization: The Missing Ingredient for Fast Stylization*. 2016. DOI: 10.48550/ARXIV.1607.08022. URL: https://arxiv.org/abs/1607.08022 (cit. on p. 3).

[15] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. *Layer Normalization*. 2016. DOI: 10.48550/ARXIV.1607.06450. URL: https://arxiv.org/abs/1607.06450 (cit. on p. 3).

[16] Dae-Young Kang, Pham Duong, and Jung-Chul Park. «Application of Deep Learning in Dentistry and Implantology». In: *The Korean Academy of Oral and Maxillofacial Implantology* 24 (Sept. 2020), pp. 148–181. DOI: 10.32542/implantology.202015 (cit. on p. 3).

[17] Sepp Hochreiter and Jürgen Schmidhuber. «Long Short-term Memory». In: *Neural computation* 9 (Dec. 1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735 (cit. on pp. 4, 38).

[18] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2014. DOI: 10.48550/ARXIV.1409.0473. URL: https://arxiv.org/abs/1409.0473 (cit. on p. 4).

[19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. *Attention Is All You Need*. 2017. DOI: 10.48550/ARXIV.1706.03762. URL: https://arxiv.org/abs/1706.03762 (cit. on pp. 4, 11).

[20] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. *Unsupervised Representation Learning by Predicting Image Rotations*. 2018. DOI: 10.48550/ARXIV.1803.07728. URL: https://arxiv.org/abs/1803.07728 (cit. on pp. 5, 6, 12, 22).

[21] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. *Unsupervised Visual Representation Learning by Context Prediction*. 2015. DOI: 10.48550/ARXIV.1505.05192. URL: https://arxiv.org/abs/1505.05192 (cit. on pp. 5, 6, 12, 23).

[22] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. *Colorization as a Proxy Task for Visual Understanding*. 2017. DOI: 10.48550/ARXIV.1703.04044. URL: https://arxiv.org/abs/1703.04044 (cit. on pp. 5, 12, 23).

[23] Richard Zhang, Phillip Isola, and Alexei A. Efros. *Colorful Image Colorization*. 2016. DOI: 10.48550/ARXIV.1603.08511. URL: https://arxiv.org/abs/1603.08511 (cit. on pp. 5, 6, 12, 23).

[24] Mehdi Noroozi and Paolo Favaro. *Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles*. 2016. DOI: 10.48550/ARXIV.1603.09246. URL: https://arxiv.org/abs/1603.09246 (cit. on pp. 5, 6, 12, 23).

[25] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. *Generative Adversarial Networks*. 2014. DOI: 10.48550/ARXIV.1406.2661. URL: https://arxiv.org/abs/1406.2661 (cit. on p. 5).

[26] Milind Soam and Sanjeev Thakur. «Next Word Prediction Using Deep Learning: A Comparative Study». In: *2022 12th International Conference on Cloud Computing, Data Science and Engineering (Confluence)*. 2022, pp. 653–658. DOI: 10.1109/Confluence52989.2022.9734151 (cit. on p. 5).

[27] Ishan Misra, C. Lawrence Zitnick, and Martial Hebert. *Shuffle and Learn: Unsupervised Learning using Temporal Order Verification*. 2016. DOI: 10.48550/ARXIV.1603.08561. URL: https://arxiv.org/abs/1603.08561 (cit. on p. 5).

[28] Metin Gurcan, Laura Boucheron, Ali Can, Anant Madabhushi, Nasir Rajpoot, and Bulent Yener. «Histopathological Image Analysis: A Review». In: *Biomedical Engineering, IEEE Reviews in* 2 (Feb. 2009), pp. 147–171. DOI: 10.1109/RBME.2009.2034865 (cit. on p. 7).

[29] Anant Madabhushi and George Lee. «Image Analysis and Machine Learning in Digital Pathology: Challenges and Opportunities». In: *Medical Image Analysis* 33 (July 2016). DOI: 10.1016/j.media.2016.06.037 (cit. on p. 7).

[30] Chetan L. Srinidhi, Ozan Ciga, and Anne L. Martel. «Deep neural network models for computational histopathology: A survey». In: *Medical Image Analysis* 67 (Jan. 2021), p. 101813. DOI: 10.1016/j.media.2020.101813. URL: https://doi.org/10.1016%5C%2Fj.media.2020.101813 (cit. on pp. 7–10).

[31] Dan C Cireşan, Alessandro Giusti, Luca M Gambardella, and Jürgen Schmidhuber. «Mitosis detection in breast cancer histology images with deep neural networks». In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2013*. Lecture notes in computer science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 411–418 (cit. on p. 7).

[32] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. 2015. DOI: 10.48550/ARXIV.1506.01497. URL: https://arxiv.org/abs/1506.01497 (cit. on p. 7).

[33] Siddhant Rao. *MITOS-RCNN: A Novel Approach to Mitotic Figure Detection in Breast Cancer Histopathology Images using Region Based Convolutional Neural Networks*. 2018. DOI: 10.48550/ARXIV.1807.01788. URL: https://arxiv.org/abs/1807.01788 (cit. on p. 7).

[34] Neeraj Kumar et al. «A Multi-Organ Nucleus Segmentation Challenge». In: *IEEE Transactions on Medical Imaging* PP (Oct. 2019), pp. 1–1. DOI: 10.1109/TMI.2019.2947628 (cit. on p. 8).

[35] Jonathan Long, Evan Shelhamer, and Trevor Darrell. *Fully Convolutional Networks for Semantic Segmentation*. 2014. DOI: 10.48550/ARXIV.1411.4038. URL: https://arxiv.org/abs/1411.4038 (cit. on p. 8).

[36] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. «U-Net: Convolutional Networks for Biomedical Image Segmentation». In: *CoRR* abs/1505.04597 (2015). arXiv: 1505.04597. URL: http://arxiv.org/abs/1505.04597 (cit. on p. 8).

[37] Nikhil Seth, Shazia Akbar, Sharon Nofech-Mozes, Sherine Salama, and Anne Martel. «Automated Segmentation of DCIS in Whole Slide Images». In: July 2019, pp. 67–74. ISBN: 978-3-030-23936-7. DOI: 10.1007/978-3-030-23937-4_8 (cit. on p. 8).

[38] Korsuk Sirinukunwattana et al. «Gland segmentation in colon histology images: The glas challenge contest». en. In: *Med. Image Anal.* 35 (Jan. 2017), pp. 489–502 (cit. on p. 8).

[39] Dongnan Liu, Donghao Zhang, Yang Song, Chaoyi Zhang, Fan Zhang, Lauren O'Donnell, and Weidong Cai. «Nuclei Segmentation via a Deep Panoptic Model with Semantic Feature Fusion». In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, July 2019, pp. 861–868. DOI: 10.24963/ijcai.2019/121. URL: https://doi.org/10.24963/ijcai.2019/121 (cit. on p. 8).

[40] Saad Wazir and Muhammad Moazam Fraz. «HistoSeg: Quick attention with multi-loss function for multi-structure segmentation in digital histology images». In: *2022 12th International Conference on Pattern Recognition Systems (ICPRS)*. IEEE, June 2022. DOI: 10.1109/icprs54038.2022.9854067. URL: https://doi.org/10.1109%5C%2Ficprs54038.2022.9854067 (cit. on p. 8).

[41] Jason Wei, Laura Tafe, Yevgeniy Linnik, Louis Vaickus, Naofumi Tomita, and Saeed Hassanpour. «Pathologist-level classification of histologic patterns on resected lung adenocarcinoma slides with deep neural networks». In: *Scientific Reports* 9 (Mar. 2019). DOI: 10.1038/s41598-019-40041-7 (cit. on p. 9).

[42] Babak Ehteshami Bejnordi et al. «Using deep convolutional neural networks to identify and classify tumor-associated stroma in diagnostic breast biopsies». In: *Modern Pathology* 31 (June 2018). DOI: 10.1038/s41379-018-0073-z (cit. on p. 9).

[43] Francesco Ponzio, Giacomo Deodato, Enrico Macii, Santa Di Cataldo, and Elisa Ficarra. «Exploiting "Uncertain" Deep Networks for Data Cleaning in Digital Pathology». In: *2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI)*. 2020, pp. 1139–1143. DOI: 10.1109/ISBI45749.2020.9098605 (cit. on p. 9).

[44] John Platt, Nello Cristianini, and John Shawe-Taylor. «Large Margin DAGs for Multiclass Classification». In: *Advances in Neural Information Processing Systems*. Ed. by S. Solla, T. Leen, and K. Müller. Vol. 12. MIT Press, 1999. URL: https://proceedings.neurips.cc/paper/1999/file/4abe17a1c80cbdd2aa241b70840879de-Paper.pdf (cit. on p. 9).

[45] Nakul Agarwal, Vineeth N Balasubramanian, and C.V. Jawahar. «Improving multiclass classification by deep networks using DAGSVM and Triplet Loss». In: *Pattern Recognition Letters* 112 (2018), pp. 184–190. ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2018.06.034. URL: https://www.sciencedirect.com/science/article/pii/S0167865518302745 (cit. on p. 9).

[46] Rudan Xiao, Eric Debreuve, Damien Ambrosetti, and Xavier Descombes. «Renal Cell Carcinoma Classification from Vascular Morphology». In: *MICCAI 2021 - 24th International Conference on Medical Image Computing and Computer Assisted Intervention*. Vol. 12906. Lecture Notes in Computer Science. Strasbourg, France: Springer International Publishing, Sept. 2021, pp. 611–621. DOI: 10.1007/978-3-030-87231-1_59. URL: https://hal.archives-ouvertes.fr/hal-03449971 (cit. on p. 9).

[47] Zeyu Gao, Pargorn Puttapirat, Jiangbo Shi, and Chen Li. «Renal Cell Carcinoma Detection and Subtyping with Minimal Point-Based Annotation in Whole-Slide Images». In: (Aug. 2020) (cit. on p. 9).

[48] Mina Khoshdeli, Alexander Borowsky, and Bahram Parvin. «Deep Learning Models Differentiate Tumor Grades from H&E Stained Histology Sections». In: *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. 2018, pp. 620–623. DOI: 10.1109/EMBC.2018.8512357 (cit. on p. 10).

[49] Karl-Friedrich Kowalewski, Luisa Egen, Chanel E. Fischetti, Stefano Puliatti, Gomez Rivas Juan, Mark Taratkin, Rivero Belenchon Ines, Marie Angela Sidoti Abate, Julia Mühlbauer, Frederik Wessels, Enrico Checcucci, and Giovanni Cacciamani. «Artificial intelligence for renal cancer: From imaging to histology and beyond». In: *Asian Journal of Urology* 9.3 (2022). Renal cancer: From current evidence to future perspectives, pp. 243–252. DOI: https://doi.org/10.1016/j.ajur.2022.05.003. URL: https://www.sciencedirect.com/science/article/pii/S221438822200042X (cit. on p. 10).

[50] Wieland Brendel and Matthias Bethge. *Approximating CNNs with Bag-of-local-Features models works surprisingly well on ImageNet*. 2019. DOI: 10.48550/ARXIV.1904.00760. URL: https://arxiv.org/abs/1904.00760 (cit. on p. 11).

[51] Michael Gadermayr and Maximilian Tschuchnig. *Multiple Instance Learning for Digital Pathology: A Review on the State-of-the-Art, Limitations & Future Potential*. 2022. DOI: 10.48550/ARXIV.2206.04425. URL: https://arxiv.org/abs/2206.04425 (cit. on p. 11).

[52] Le Hou, Dimitris Samaras, Tahsin M. Kurc, Yi Gao, James E. Davis, and Joel H. Saltz. *Patch-based Convolutional Neural Network for Whole Slide Tissue Image Classification*. 2015. DOI: 10.48550/ARXIV.1504.07947. URL: https://arxiv.org/abs/1504.07947 (cit. on p. 11).

[53] Zhipeng Jia, Xingyi Huang, Eric I-Chao Chang, and Yan Xu. «Constrained Deep Weak Supervision for Histopathology Image Segmentation». In: *IEEE Transactions on Medical Imaging* 36.11 (Nov. 2017), pp. 2376–2388. DOI: 10.1109/tmi.2017.2724070. URL: https://doi.org/10.1109%5C%2Ftmi.2017.2724070 (cit. on p. 11).

[54] Maximilian Ilse, Jakub M. Tomczak, and Max Welling. *Attention-based Deep Multiple Instance Learning*. 2018. DOI: 10.48550/ARXIV.1802.04712. URL: https://arxiv.org/abs/1802.04712 (cit. on p. 11).

[55] Richard J. Chen, Chengkuan Chen, Yicong Li, Tiffany Y. Chen, Andrew D. Trister, Rahul G. Krishnan, and Faisal Mahmood. *Scaling Vision Transformers to Gigapixel Images via Hierarchical Self-Supervised Learning*. 2022. DOI: 10.48550/ARXIV.2206.02647. URL: https://arxiv.org/abs/2206.02647 (cit. on pp. 11, 12).

[56] Navid Alemi Koohbanani, Balagopal Unnikrishnan, Syed Ali Khurram, Pavitra Krishnaswamy, and Nasir Rajpoot. *Self-Path: Self-supervision for Classification of Pathology Images with Limited Annotations*. 2020. DOI: 10.48550/ARXIV.2008.05571. URL: https://arxiv.org/abs/2008.05571 (cit. on pp. 12, 13, 24, 25, 28).

[57] Chetan L. Srinidhi, Seung Wook Kim, Fu-Der Chen, and Anne L. Martel. «Self-supervised driven consistency training for annotation efficient histopathology image analysis». In: *Medical Image Analysis* 75 (Jan. 2022), p. 102256. DOI: 10.1016/j.media.2021.102256. URL: https://doi.org/10.1016%5C%2Fj.media.2021.102256 (cit. on pp. 13, 15, 24, 25, 29, 34, 61).

[58] Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V. Le. *Unsupervised Data Augmentation for Consistency Training*. 2019. DOI: 10.48550/ARXIV.1904.12848. URL: https://arxiv.org/abs/1904.12848 (cit. on p. 14).

[59] Joseph Boyd, Mykola Liashuha, Eric Deutsch, Nikos Paragios, Stergios Christodoulidis, and Maria Vakalopoulou. *Self-Supervised Representation Learning using Visual Field Expansion on Digital Pathology*. 2021. DOI: 10.48550/ARXIV.2109.03299. URL: https://arxiv.org/abs/2109.03299 (cit. on pp. 14, 15).

[60] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. *Adversarial Autoencoders*. 2015. DOI: 10.48550/ARXIV.1511.05644. URL: https://arxiv.org/abs/1511.05644 (cit. on p. 14).

[61] Ruiwen Ding, Anil Yadav, Erika Rodriguez, Ana Cristina Araujo Lemos da Silva, and William Hsu. «Improving the Self-Supervised Pretext Task for Histopathologic Subtype Classification». In: *Medical Imaging with Deep Learning*. 2022. URL: https://openreview.net/forum?id=7QWzEwByMXq (cit. on pp. 14, 15, 24, 25, 28, 34, 53, 56, 58).

[62] Nikki S Vyas, Michael Markow, Carlos Prieto-Granada, Sudeep Gaudi, Leslie Turner, Paul Rodriguez-Waitkus, Jane L Messina, and Drazen M Jukic. «Comparing whole slide digital images versus traditional glass slides in the detection of common microscopic features seen in dermatitis». en. In: *J. Pathol. Inform.* 7.1 (July 2016), p. 30 (cit. on p. 16).

[63] Nehal Atallah, Michael Toss, Clare Verrill, Manuel Salto-Tellez, David Snead, and Emad Rakha. «Potential quality pitfalls of digitalized whole slide image of breast pathology in routine practice». In: *Modern Pathology* 35 (Dec. 2021). DOI: 10.1038/s41379-021-01000-8 (cit. on p. 17).

[64] Mark D. Zarella, Douglas Bowman; Famke Aeffner, Navid Farahani, Albert Xthona; Syeda Fatima Absar, Anil Parwani, Marilyn Bui, and Douglas J. Hartman. «A Practical Guide to Whole Slide Imaging: A White Paper From the Digital Pathology Association». In: *Archives of Pathology & Laboratory Medicine* 143.2 (Oct. 2018), pp. 222–234. ISSN: 0003-9985. DOI: 10.5858/arpa.2018-0343-RA. URL: https://doi.org/10.5858/arpa.2018-0343-RA (cit. on p. 18).

[65] Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. «Multimodal Machine Learning: A Survey and Taxonomy». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41.2 (2019), pp. 423–443. DOI: 10.1109/TPAMI.2018.2798607 (cit. on p. 27).

[66] Ross Girshick. *Fast R-CNN*. 2015. DOI: 10.48550/ARXIV.1504.08083. URL: https://arxiv.org/abs/1504.08083 (cit. on p. 32).

[67] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. *Highway Networks*. 2015. DOI: 10.48550/ARXIV.1505.00387. URL: https://arxiv.org/abs/1505.00387 (cit. on p. 38).

[68] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. *Training Very Deep Networks*. 2015. DOI: 10.48550/ARXIV.1507.06228. URL: https://arxiv.org/abs/1507.06228 (cit. on p. 38).

[69] Priya Goyal, Piotr Dollár, Ross B. Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. «Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour». In: *CoRR* abs/1706.02677 (2017). arXiv: 1706.02677. URL: http://arxiv.org/abs/1706.02677 (cit. on p. 47).

[70] Samuel L. Smith, Pieter-Jan Kindermans, and Quoc V. Le. «Don't Decay the Learning Rate, Increase the Batch Size». In: *CoRR* abs/1711.00489 (2017). arXiv: 1711.00489. URL: http://arxiv.org/abs/1711.00489 (cit. on p. 47).

[71] Ronald Kemker, Marc McClure, Angelina Abitino, Tyler Hayes, and Christopher Kanan. «Measuring Catastrophic Forgetting in Neural Networks». In: *Proceedings of the AAAI Conference on Artificial Intelligence* 32.1 (Apr. 2018). DOI: 10.1609/aaai.v32i1.11651. URL: https://ojs.aaai.org/index.php/AAAI/article/view/11651 (cit. on p. 47).

[72] Robert M. French. «Catastrophic forgetting in connectionist networks». In: *Trends in Cognitive Sciences* 3.4 (1999), pp. 128–135. ISSN: 1364-6613. DOI: https://doi.org/10.1016/S1364-6613(99)01294-2. URL: https://www.sciencedirect.com/science/article/pii/S1364661399012942 (cit. on p. 47).

[73] Matthew Jagielski, Om Thakkar, Florian Tramèr, Daphne Ippolito, Katherine Lee, Nicholas Carlini, Eric Wallace, Shuang Song, Abhradeep Thakurta, Nicolas Papernot, and Chiyuan Zhang. *Measuring Forgetting of Memorized Training Examples*. 2022. DOI: 10.48550/ARXIV.2207.00099. URL: https://arxiv.org/abs/2207.00099 (cit. on p. 47).

[74] Ilya Loshchilov and Frank Hutter. *SGDR: Stochastic Gradient Descent with Warm Restarts*. 2016. DOI: 10.48550/ARXIV.1608.03983. URL: https://arxiv.org/abs/1608.03983 (cit. on p. 48).

[75] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. *Distilling the Knowledge in a Neural Network*. 2015. DOI: 10.48550/ARXIV.1503.02531. URL: https://arxiv.org/abs/1503.02531 (cit. on p. 48).

[76] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. «Curriculum Learning». In: ICML '09. Montreal, Quebec, Canada: Association for Computing Machinery, 2009, pp. 41–48. ISBN: 9781605585161. DOI: 10.1145/1553374.1553380. URL: https://doi.org/10.1145/1553374.1553380 (cit. on p. 48).

[77] Xin Wang, Yudong Chen, and Wenwu Zhu. «A survey on curriculum learning». en. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 44.9 (Sept. 2022), pp. 4555–4576 (cit. on p. 48).

[78] Petru Soviany, Radu Tudor Ionescu, Paolo Rota, and Nicu Sebe. *Curriculum Learning: A Survey*. 2021. DOI: 10.48550/ARXIV.2101.10382. URL: https://arxiv.org/abs/2101.10382 (cit. on p. 48).

[79] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. «Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization». In: *International Journal of Computer Vision* 128.2 (Oct. 2019), pp. 336–359. DOI: 10.1007/s11263-019-01228-7. URL: https://doi.org/10.1007%2Fs11263-019-01228-7 (cit. on p. 49).

[80] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. *Learning Deep Features for Discriminative Localization*. 2015. DOI: 10.48550/ARXIV.1512.04150. URL: https://arxiv.org/abs/1512.04150 (cit. on p. 49).