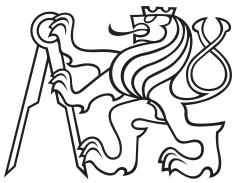
Bachelor Project



Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Methods for hierarchical classification of histopathology images

Aleš Král

Supervisor: prof. Dr. Ing. Jan Kybic
Field of study: Open informatics
May 2022

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Král Aleš** Personal ID number: **492163**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Open Informatics**

Specialisation: **Artificial Intelligence and Computer Science**

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Methods for Hierarchical Classification of Histopathology Images

Bachelor’s thesis title in Czech:

Metody pro hierarchickou klasifikaci histopatologických obrazů

Guidelines:

Histopathological images are too large to be processed at once by standard deep-learning approaches and reducing their resolution leads to a reduced performance. The goal of this thesis is to implement and experimentally compare several hierarchical image classification methods and compare their performance on the CAMELYON16 and 17 datasets.
Instructions:
1. Get familiar with the datasets and existing methods for solving this task.
2. Re-implement, tune and test the following methods, possibly based on existing implementations. Describe these methods, focusing on their common points and differences.
  Reinforcement learning approach [1].
  Quadtree approach [2].
  Recurrent visual attention model approach [3].
  Attention-based deep multiple instance learning [4].
  Clustering-constrained attention multiple instance learning (CLAM) approach [5].
3. Evaluate the performance of the studied methods, especially their speed, accuracy and training requirements.
4. Choose the best performing method, suggest an improvement, implement and test it.

Bibliography / sources:

[1] Hering J. and Kybic J.. "Multiple Instance Learning Via Deep Hierarchical Exploration for Histology Image Classification." ISBI: International Symposium Biomedical Imaging, pp. 235-238, 2020.
[2] Jewsbury, R., Bhalerao, A. and Rajpoot, N.M. “A QuadTree Image Representation for Computational Pathology”. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 648-656. 2021.
[3] BenTaieb, Aïcha, and Ghassan Hamarneh. “Predicting Cancer with a Recurrent Visual Attention Model for Histopathology Images.” In Proc. of MICCAI 2018, pp 127-137, 2018.
[4] Ilse, M., Tomczak, J. M., Welling, M. (2018). “Attention-based Deep Multiple Instance Learning”. International Conference on Machine Learning, Stockholm, Sweden, PMLR, pp 2127-2136, 2018.
[5] Lu, Ming Y., et al. “Data-efficient and weakly supervised computational pathology on whole-slide images”. Nature Biomedical Engineering vol 5, pp 555–570. 2020.

Name and workplace of bachelor's thesis supervisor:

prof. Dr. Ing. Jan Kybic Biomedical imaging algorithms FEE

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **08.02.2022** Deadline for bachelor thesis submission: **20.05.2022**

Assignment valid until: **30.09.2023**

prof. Dr. Ing. Jan Kybic	prof. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor's signature	Head of department's signature	Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

Acknowledgements

First and foremost, I would like to sincerely thank my supervisor prof. Dr. Ing. Jan Kybic, for his great guidance and feedback throughout the making of this thesis. He always promptly responded when I ran into trouble and gave valuable comments and advice.

I wish to express my deepest gratitude towards my family and friends who provided me with encouragement throughout the study years.

Finally, I would like to thank the Czech Technical University in Prague for the education provided.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

In Prague, May 20, 2022

Abstract

In recent years, many methods for hierarchical image classification have been developed showing great promises in the automation of histology image analysis. In this thesis, we implemented some existing methods, mainly the Quadtree approach and the Recurrent visual attention model, for the detection of metastasis in sentinel lymph nodes and compared their performance according to various criteria. We propose modifications to the existing algorithms which improve their ability to correctly distinguish between tumorous and normal tissue.

We also show and utilize different tools and techniques for the preprocessing of whole slide images, namely the patch extraction of the Clustering-constrained attention multiple instance learning method or the Quadtree patch extraction.

The implemented methods have been compared between each other and their references by providing them with test data from the CAMELYON16 challenge. The results were satisfactory for the classification of tiles in whole slide images, with accuracy converging at 87% for the Recurrent visual attention model and 92% for the Quadtree method reaching reference accuracy and sometimes surpassing it. On the other hand, best slide classification from our implementations was around 81% from the Quadtree method and accuracy of the tested CLAM method was around 99% surpassing all other models.

Keywords: machine learning, breast cancer, hierarchical classification, neural network, pathology

Supervisor: prof. Dr. Ing. Jan Kybic
Katedra kybernetiky
FEE CTU
Technická 2,
Praha 6

Abstrakt

V posledních letech bylo vyvinuto mnoho metod pro hierarchickou klasifikaci obrazu, které vykazují uplatnění v oblasti automatizace analýzy histologických obrazů. V této práci jsme implementovali některé existující metody, především Quadtree a Recurrent visual attention model, pro detekci metastáz v sentinelových lymfatických uzlinách a porovnali jejich výkonnost na základě různých kritérií. Navrhujeme úpravy stávajících algoritmů, které zlepšují jejich schopnost správně rozlišit mezi nádorovou a normální tkání.

Ukazujeme a využíváme také různé nástroje a techniky pro předzpracování snímků celých preparátů, konkrétně extrakci výřezků Clustering-constrained attention multiple instance learning metodou nebo metodou Quadtree.

Implementované metody byly porovnány mezi sebou a svými referencemi tak, že jim byla poskytnuta testovací data ze soutěže CAMELYON16. Výsledky byly uspokojivé pro klasifikaci dlaždic na snímcích celých preparátů, přičemž přesnost konvergovala k 87% u Recurrent visual attention modelu a u metody Quadtree k 92% kde dosahovala referenční přesnosti a někdy ji překonávala. Na druhou stranu nejlepší klasifikace snímků celých preparátů z našich implementací byla přibližně 81% u metody Quadtree a přesnost testované metody CLAM byla přibližně 99% která překonala všechny ostatní modely.

Klíčová slova: strojové učení, rakovina prsu, hierarchická klasifikace, neuronové sítě, patologie

Překlad názvu: Metody pro hierarchickou klasifikaci histopatologických obrazů

Contents

Figures

Tables

Abbreviations and acronyms

WSI	**W**hole **S**lide **I**mage
RVAM	**R**ecurrent **V**isual **A**ttention **M**odel
CAMELYON16	**CA**ncer **ME**tastasis in **LY**mph n**O**des challe**N**ge 2016
AUC	**A**rea **U**nder **C**urve
GB	**G**iga **B**yte
TIF	**T**agged **I**mage **F**ormat
SVS	**G**iga **B**yte
JPEG	**J**oint **P**hotographic **E**xperts **G**roup
TCGA	**T**he **C**ancer **G**enome **A**tlas
SVM	**S**upport **V**ector **M**achines
ResNet	**Res**idual neural **Net**work
H&E	**H**aematoxylin and **E**osin
MIL	**M**ultiple **I**nstance **L**earning
CLAM	**CL**ustering-constrained **A**ttention **M**ultiple instance learning
ReLU	**R**ectified **L**inear **U**nit
RUMC	**R**adbound **U**niversity **M**edical **C**enter
UMCU	**U**niversity **M**edical **C**enter **U**trecht
ROC	**R**eceiver **O**perator **C**haracteristic
AUC-ROC	**A**rea **U**nder the **R**eceiver **O**perator **C**haracteristic **C**urve
TPR	**T**rue **P**ositive **R**ate
DAB	**D**i**A**mino**B**enzidine
FPR	**F**alse **P**ositive **R**ate

Chapter 1

Introduction

Breast cancer is one of the main causes of cancer worldwide. Early diagnostics significantly increase the chances of correct treatment and survival.

In the process of histology image analysis for cancer diagnosis, pathologist standardly observes the tissue, its distribution and regularities in cell shapes. Because the tissue slides, called WSIs (Whole Slide Images), are so complex and large in scale (often over 2GB), correct classification of tissue whether it is benign or malignant is often very time-consuming. Fortunately computer-aided analysis has become a rapidly expanding field within the past decade. Computerised scans of stained tissue slides are used by various algorithms to automate tissue classification and aid pathologists.

1.1 Motivation

In the recent past, methods for hierarchical image classification have become quite popular with researchers from different countries tackling on the problem of WSI image classification using various resources and approaches. In practice, these are valuable to automate classification for example of cancer metastasis in lymph nodes of breasts [3] or colon cancer [15]

Most approaches[8][14] try to solve this problem by applying real world practice. WSIs have various staining methods applied to them, most com-

monly the usage of haematoxylin and eosin (H&E) staining can be observed. This gives the observer, whether it is a human or a machine, information about the most discriminate regions of a tissue slide.

Same as a human analyst, computers have access to different parts of the images at different scales and resolutions. This image decomposition is called the multi-resolution pyramid [11]. It lets researchers use tools for manipulation with these "pyramids" and write algorithms for automatic passing through different levels of magnification and extract information used for image classification.

Many such methods have been developed in recent years, with a difference in performances between each one. Comparison between such methods is crucial, since a researcher or a medical expert could be in search of automatising tissue classification. Having so many options online at our disposal, finding the optimal approach and doing its re-implementation could often be time-consuming.

1.2 Goals

The main focus of this work is to compare different methods for hierarchical image classification, preferably on the same datasets.

In the following chapters some methods for hierarchical image classification will be introduced. Some focus on tissue segmentation to extract only the most informative regions, others try to develop state-of-art deep neural networks which aggregate large quantities of extracted patches to correctly classify given images.

While some methods may be already implemented and used for demonstration [27], others [15][3] will be implemented according to the architecture given to us by the papers. Some articles may not be thorough in the architecture specifications. In these cases, we will try adding modifications, which have improved other models in the past. In the final chapters referenced and modified versions will be discussed for their ability to correctly classify tissue images and how they are able to fulfill their proposed functions.

Chapter 2

Background

2.1 Introduction

Before we begin explaining each method, we first need to get an understanding about tools we will be working with while constructing and testing different approaches. Better understanding of terms explained in this chapter will help us deal with the task.

2.2 Whole slide imaging

Also commonly referred to as "virtual" microscopy [19] involves the scanning of glass slides to produce digital slides. These systems offer pathologists an alternate mechanism to menage and interpret information.

The virtualization is commonly achieved by capturing many small high resolution image tiles and then stitching them together to create a full image for histology analysis. The sequential parts [21] which make the process of digitisation are: image acquisition, storage, editing and display of images.

- Scanners

Whole slide image scanners are specialized devices that are dedicated to acquiring high resolution images of entire slides. These slide scanners consist of 4 main components: light source, slide stage, objective lenses and a high resolution camera for image capture. Whole slide scanners capture images of tissue tile by tile and then "stitch" them to create a digital image of the entire slide.

- WSI

  The slide is standardly captured at high resolutions. Common being x20 or x40 magnification. The resulting structure of the WSI is then called a multi-resolution pyramid.

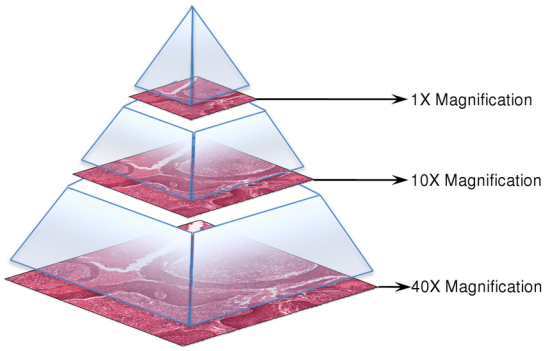


Figure 2.1: Multi-resolution pyramid showing the same WSI image on different levels of magnification taken from the CAMELYON16 challenge [2]

  Resolution of a WSI is typically expressed in μm per pixel. A typical whole slide image scanned at x40 magnification has a resolution of about 0.25 μm per pixel with a 24-bit color depth [21].

- Storage

  Acquiring a decently sized dataset leads to huge storage requirements. Since compression of scanned images leads to loss of information and resolution of the image, WSIs are loaded into lossless [25] image formats to preserve data. Lossy data format would be something like JPEG since it compresses the files to a smaller size, but at the cost of data loss, whereas lossless data-formats preserve the information at the cost of a large size. The most common formats used in Medical imaging being .tif or .svs.

2.3 Image tools

Working with gigapixel images is a challenging if not impossible task for standard tools since these are typically designed for images that can be

comfortably uncompressed into RAM or a swap file, so alternate methods need to be utilized.

2.3.1 Openslide

The library Openslide for python [10] lets the user load a WSI as an object Openslide. With this object, OpenSlide allows reading a small amount of image data at the resolution closest to a desired zoom level. This means that the magnification level, coordinates and scale can be specified to read a region from a WSI. Other functions let us save the thumbnail of the original image for example.

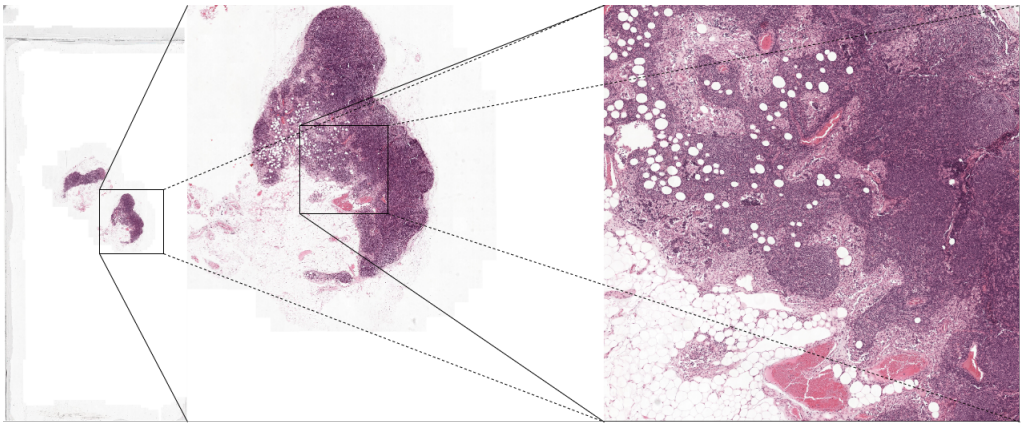


Figure 2.2: The usage of Openslide library on a wsi image. From left to right: thumbnail of the original WSI at the highest zoom level (lowest resolution), extracted region at a zoom level 3, extracted region at a zoom level 1

2.4 Preprocessing techniques

Preprocessing of WSIs is crucial, since it is impossible for a classification model to work with a full resolution whole slide image. It needs to attend to extracted "patches" or "tiles" and aggregate individual results to form a slide-level classification.

Usually, we first want to eliminate the white background, which doesn't contain any tissue and find a mask that highlights areas containing the tissue. This is done by thresholding mechanisms [23], while the most popular being Otsu's [17].

2.4.1 Otsu's thresholding algorithm

This technique scans all possible threshold values and tries to find the optimal one. It assumes that the image consists of only two channels (gray scale). A value between two peaks in the pixel distribution is calculated as the thresholding point. Using the this point, the foreground is separated from the background.

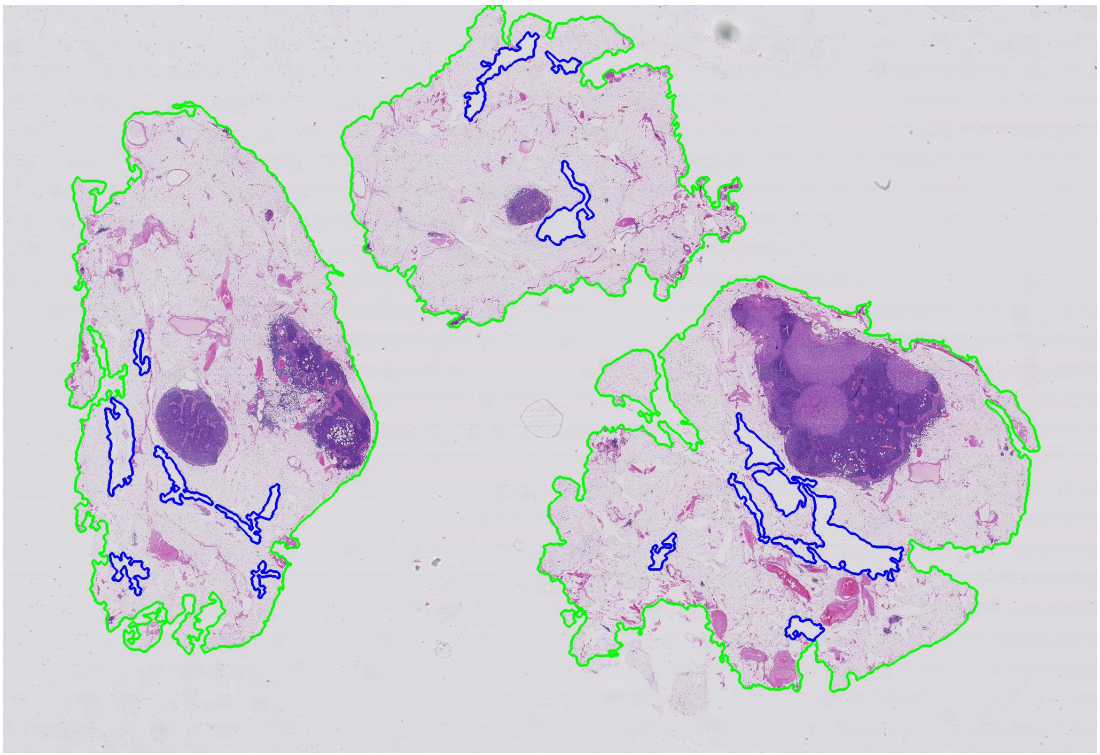


Figure 2.3: Contours around tissue areas, which have been chosen by a thresholding algorithm (green) and countours of holes, which do not contain tissue (blue) [27]

The size of the foreground region is much smaller, but it is still not possible to be used as it is. This part varies from method to method, but generally tiles or patches need to be extracted from the entire foreground to produce a dataset for each WSI. Some methods utilize larger tiles [3] (approx. 5000×5000 pixels), while other smaller patches [27] (approx. 256×256 pixels)

2.5 MIL

MIL is a variation of supervised learning where a single class label is assigned to a bag of instances [14]. This model falls under the supervised learning framework, where every instance is assigned a class label, even if artificial. The bag is then positively labeled if at least one instance in it is positive and negatively if all instances in it are negative. Here, I will talk about a specific implementation of the MIL problem in the setting of machine learning.

In [14] they propose a trainable and not pre-defined MIL pooling operator. The biggest advantage is that it can be translated to a neural network setting called **attention mechanism**.

In the attention mechanism, weighed average across all instances (low-dimensional embeddings) is used where the weights are determined by layers of a neural network. Let $H = \{h_1, ..., h_K\}$ be a bag of K instances. The MIL pooling operator will then look like this:

$$z = \sum_{k=1}^{K} a_k h_k \tag{2.1}$$

where:

$$a_k = \frac{\exp\{w^T \tanh(V h_k^T)\}}{\sum_{j=1}^{K} \exp\{w^T \tanh(V h_j^T)\}} \tag{2.2}$$

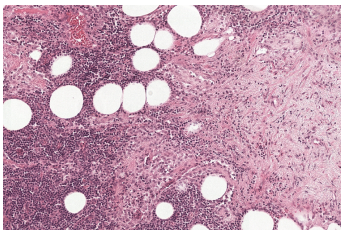
where $w \in \mathbb{R}^{L\times 1}$ and $V \in \mathbb{R}^{L\times M}$ are parameters.

2.6 Tissue staining

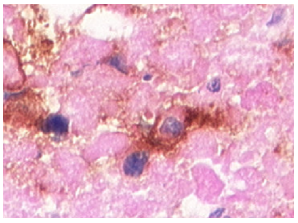
The explanation of tissue staining will help us in understanding some parts of methods for hierarchical image classification. It is a vital part of image digitization and helps to bring contrast and color to tissue by facilitating a chromatic distinction among different tissue components [6].

The most common stain is the haematoxylin and eosin, also known as H&E. It is widely used, because it is comprised of compounds which are much

cheaper than other stains. That is why it is sometimes referred to as a routine stain. Because of its relative ease of use, H&E stain is applied to nearly all clinical cases covering cca. 80% of all human tissue stained globally[1]. There is also a wide variety of other staining compounds, which are used in different situations to highlight different constituents of tissue. One such compound is diaminobenzidine (DAB) that is used for the staining of nucleic acids and proteins [12]. DAB, which is brown in color, is sometimes used with the combination of haematoxylin, which is blue, mainly staining the cell nuclei, and eosin, magenta-red, acting as a cytoplasmic stain.



(a) : H&E stain



(b) : DAB and H&E stain

Figure 2.4: Comparison of different stains

Chapter 3

Methods background

In this chapter, I will give an overview of the methods I am going to bee implementing (Recurrent visual attention network[3] and Quadtree approach [15]), which are described in detail in the following chapters and methods I will use in my work as a reference for the results (CLAM [13]).

Since these methods deal with hierarchical image classification, it is first worth explaining what that entails. In the case of WSI image classification, smaller images at various scales are extracted from the entire slides to form batches of inputs to the classification networks. This means, that the methods are trained on tiles or patches given the context of the entire slide and relation between them. The results from each tile/patch classification are aggregated to form a prediction for the WSI classification.

3.1 Recurrent visual attention network

This method [3] proposes to use a recurrent visual attention network to extract a sequence of glimpses by dynamically selecting a sequence of locations at each timestep. The recurrent part of the network is able to predict the slide level label by keeping a hidden state of the network, which aggregates information from past glimpses. Only the extracted regions bound by their locations are evaluated by convolutional neural networks and their parts to predict metastases in sentinel lymph nodes.

3.1.1 Network overview

Overview of the network which I used as part of my implementation taken and modified from [3]:

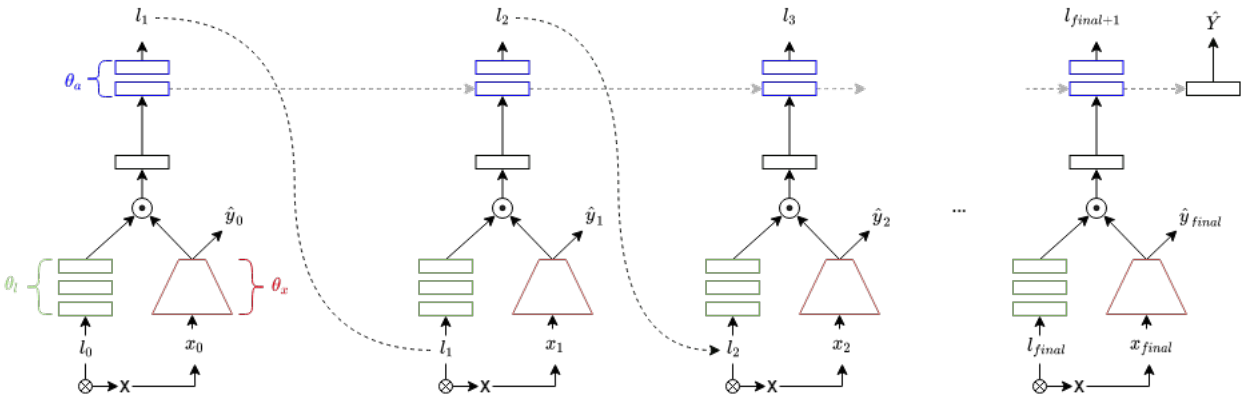


Figure 3.1: Recurrent visual attention model. Rectangular boxes represent fully connected layers and trapezoid boxes represent a CNN network. Here $\mathbf{X}$ represents the tissue slide. θ_l, θ_x and θ_a are the model parameters. $\{\hat{y}_0, \hat{y}_1, ..., \hat{y}_{final}\}$ are the predicted labels for each extracted patch and $\hat{Y}$ is the predicted slide-level label. $\{l_0, l_1, ..., l_{final}\}$ are the locations at which the network extracts the patches $\{x_0, x_1, ..., x_{final}\}$. Grey dashed lines represent the hidden state of the network. Dashed lines represent the flow of information from from one timestep to another. Black lines represent the flow of information inside the network. $\odot$ is the Hadamard product and $\otimes$ represents a matrix multiplication.

The entire network essentially starts by initializing a hidden state h_t, which serves as the aggregator of information through the passes of the network. The first location l_t is initialized to represent the entire input image X. l_t extracts a patch x_t at a timestep t and feeds the location into a location network θ_l and the x_t into network θ_x. The θ_l extracts the feature representation of l_l and θ_x creates a feature representation of x_t. These spatial and appearance feature representations are then aggregated to form a representation of a glimpse g_t by a piece-wise multiplication of the two feature vectors. The g_t is then aggregated with the previous h_{t-1} to form the next hidden state h_t in the attention network θ_a. The next location l_{t+1} is formed from the new h_t. After all glimpses have been aggregated a final hidden state h_T, where T is the maximum number of glimpses, is passed through a classification network (which is just one fully connected layer) to produce the slide-level label prediction.

3.1.2 Objective loss

The authors proposed to use a hybrid loss on which the network should be trained, comprised of four sub-losses. The main slide-level loss guides the attention network θ_a to correctly classify the aggregated glimpses in the final hidden state h_T. Next a patch-level loss is introduced to guide feature-extraction mechanism θ_x. Finally, two other losses are used. One called by the authors as the attention loss, which encourages the system to gradually approach the most discriminate areas of tissue by giving higher penalties for incorrectly predicted patches and the other called the location loss, which encourages exploration by enforcing large differences between probabilities.

For a thorough description of the implementation of this method I refer the reader to the next chapter 4.2 and the original method [3].

3.2 Quadtree approach

This method proposes an original way of data preprocessing, which tries to eliminate parts which are certainly not foreground of an image and focus on those that seem to be foreground. The authors used a data structure called quadtree [15], which is comprised of nodes that each have four children and hold information about the location they attend to in the image. In this work they proved that the Quadtree segmentation yields great results, by using less data and efficient classification.

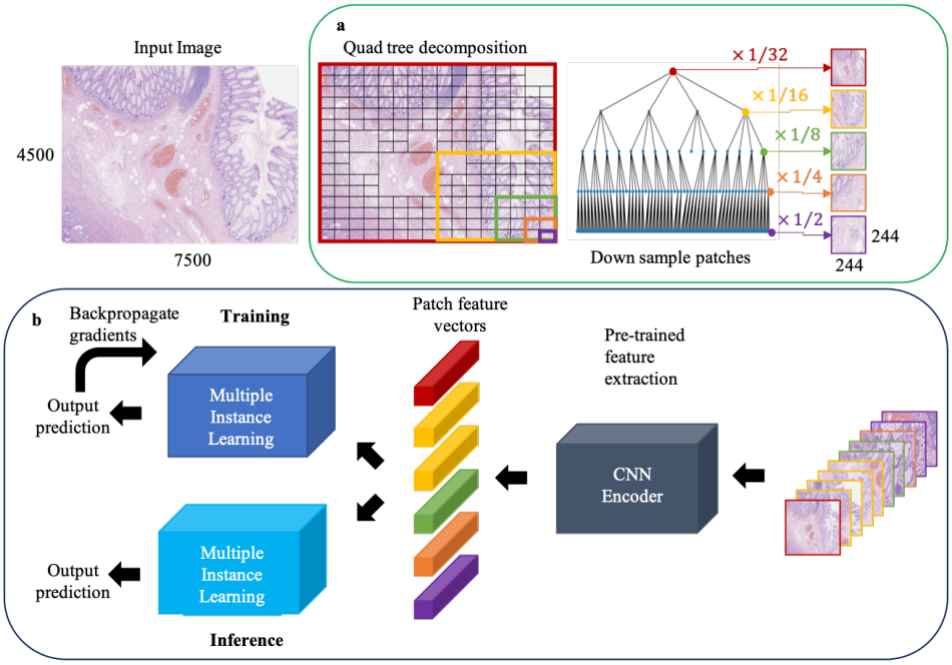


Figure 3.2: Proposed framework of the Quadtree approach, taken from [15]

This is the framework proposed in [15]. The pipeline is comprised of Quadtree patch extraction, followed by a pre-trained CNN, which extracts features from the patches and then a MIL classifier is used for image-level classification to obtain a prediction for the image-level label.

Quadtree segmentation essentially starts with the context of the entire image and then is split into four sub-quadrants of the same size. A criterion and a threshold determine whether there is enough sufficient information in the quadrant and if it should be split further or will be considered a leaf node. A quadrant is declared a leaf node if its splits would be smaller than the chosen minimum pixel size. After there are no more sub-quadrants to split, the algorithm extracts patches defined by the leaf node splits and resizes them into a fixed pixel size.

The extracted patches are then sent into a pretrained CNN, which extracts feature representations of each patch. The authors then used two approaches. One was to pass the feature vectors into a MIL 2.5 classifier and the other to use the MIL extension, CLAM, which I also used in my implementation 4.3 of this method. I will be explaining the CLAM method in the next subsection.

3.3 CLAM

Clustering-constrained Attention Multiple Instance Learning (CLAM) is a high-throughput framework, which builds upon the Multiple Instance Learning model. It was designed to solve image classification problems in computational pathology in a weakly-supervised setting. This means, that it accounts for having available only a WSI and a slide-level label. No pixel-level or region specific annotations are present.

The MIL algorithm presented in the Quadtree approach was limited to only predicting 2 class labels, positive and negative. In contrast, CLAM is applicable to general multi-class classification problems, but it should also outperform other algorithms designed for a 2-class task, which is suitable for our problem of binary classification.

3.3.1 Data preprocessing

The implementation of the CLAM network from [27] has its own data preprocessing algorithm, which I used for testing purposes. The goal of this algorithm is to prepare patches of fixed size (256×256 in the reference) and pass them through a feature extraction network to produce feature vectors for the classification network. The patches are extracted by first removing the background of a WSI and creating contours around the foreground. Then, the extracted foreground is split into thousands of fixed size patches. These patches are then sent through a pretrained CNN to extract features for each patch.

3.3.2 Classification network

The CLAM network, which classifies the patch features, was built upon the trainable attention mechanism introduced in 2.5. In a multi-class classification problem, the attention network predicts n sets of attention scores to the n classes. This lets the network learn independently for each class, which features should be considered characteristic and which uninformative for the class. In our case, we consider only 2 classes, since we deal with a binary classification problem. This part will be explained thoroughly, since this

mechanism is used in my implementation to classify results from the Quadtree segmentation 4.3.

Architecture

The first component of the network is a fully connected layer $W_1 \in \mathbb{R}^{512\times1024}$, which translates the fixed-dimensional patch-level representation z_k (where z_k is 1024-dimensional from the output of the pretrained feature extractor) into a lower-dimensional embedding h_k:

$$h_k = W_1 z_k^T \tag{3.1}$$

Now, lets consider two layers of the attention network $U_a \in \mathbb{R}^{256\times512}$ and $V_a \in \mathbb{R}^{256\times512}$ as an attention backbone of the algorithm, which is shared by all classes and lower-dimensional embeddings h_k serve as an input to them. Apart from the default attention mechanism 2.5, here we are going to have n parallel attention branches $W_{a,1}, ..., W_{a,n} \in \mathbb{R}^{1\times256}$ and similarly n parallel independent classifiers $W_{c,1}, ..., W_{c,n}$, which score each slide-specific slide-level representation. In the case of this work n is 2 for all parts of the network, since we deal with only two classes. Here, the authors present a more general problem, which could be applied to a multi-class classification task. The modified attention mechanism is then defined as:

$$h_{slide,m} = \sum_{k=1}^{N} a_{k,m} h_k \tag{3.2}$$

where:

$$a_{k,m} = \frac{\exp\{W_{a,m}(\tanh(V_a h_k^T) \odot \sigma(U_a h_k^T))\}}{\sum_{j=1}^{K} \exp\{W_{a,m}(\tanh(V_a h_j^T) \odot \sigma(U_a h_j^T))\}} \tag{3.3}$$

Where $a_{k,m}$ denotes the attention scores for the k^{th} patch and m^{th} class. σ represents the sigmoid activation layer and $h_{slide,m}$ is the aggregated slide-level representation. Furthermore, the slide-level representation is passed to the classifiers $W_{c,1}, ..., W_{c,n}$ that output the slide-level score $s_{slide,m}$.

To further encourage learning for class-specific features, **Instance-level clustering** is introduced. The point of the clustering is that the objective function (i. e. the loss) is optimized over the subset of patches, which the model either strongly attends to or completely ignores. n hidden units of the clustering network $W_{inst,m} \in \mathbb{R}^{2\times512}$ are placed after the first layer W_1 for

each class n. This gives us the assignment scores $p_{m,k}$ for every class and instance.

$$p_{m,k} = W_{inst,m} h_k^T \tag{3.4}$$

Clustering network

The outputs of the attention network are used to supervise the clustering by creating pseudolabels y for every attended instance. The pseudolabels are created in the following manner: First, we refer to the attention branch, which corresponds with the current slide-level ground truth class Y, $W_{a,Y}$, as "in-the-class" and the remaining class branches as "out-of-the-class". Then, for every in-the-class label, the attention scores $(a_{1,in-the-class}, ..., a_{K,in-the-class})$ are sorted in an ascending order and the pseudolabels for in-the-class cluster are set negative (meaning negative evidence i. e. not informative for the current class) for the first B lowest attention scores and positive (meaning positive evidence i. e. informative for the current class) for the $B+k$ attention scores (k being number of instances/patches). Then predictions for negative and positive labeled instances are created $p_{m,b}$ and $p_{m,b+B}$, where b is in the range from 1 to B. Predictions for both are created using the hidden units of the clustering network $W_{inst,m}$ applied to the individual sorted instances $\tilde{h}_1, ..., \tilde{h}_K$

For the other out-of-the-class classes, which are referred to as mutually exclusive, the list of attention scores is again sorted from the lowest score to the highest $(a_{1,out-of-the-class}, ..., a_{K,out-of-the-class})$. The first B pseudolabels are then labeled as negative (here referred to as false positive evidence) and predictions for the false positives are $p_{m,b}$, which are computed in the same way as in the in-the-class part.

The output of this algorithm is then essentially all in-the-class clustering predictions and only selected out-of-the-class predictions. The value of B is not explicitly defined in the paper, but everything suggests that it is a hyperparameter, which is chosen by the user. The entire clustering mechanism is shown in pseudocode below.

Algorithm 1: Instance-level Clustering

Function CLUSTER$((h_1, a_1), \ldots, (h_K, a_K), Y)$

- **for** $m \leftarrow$ *1,2,...,n* **do**
  - **if** $m == Y$ **then**
    - $\left(\tilde{h}_1, \tilde{a}_{1,m}\right), \ldots, \left(\tilde{h}_K, \tilde{a}_{K,m}\right)$ =SortAscending$((h_1, a_{1,m}), \ldots, (h_K, a_{K,m}))$
    - **for** $b \leftarrow$ *1,...,* B **do**
      - *generate pseudo label for positive and negative evidence*
      - $y_{m,b} = 0$
      - $y_{m,b+B} = 1$
      - *cluster assignment prediction*
      - $P_{m,b} = W_{inst,m}\tilde{h}_b^T$
      - $P_{m,b+B} = W_{inst,m}\tilde{h}_{K-B+b}^T$
  - **else**
    - *pass*
- **return** $[P_Y], [y_Y]$

This algorithm was taken from the original article [27] and modified to contain only the parts I will be using in my implementation. This was done because we don't deal with a sub-typing problem, but only a binary classification task meaning we output all in-the-class predictions and ignore all out-of-the-class predictions.

Chapter 4

Methods implementation

4.1 Introduction

This chapter aims to explain the implementation of two of the methods Quadtree approach[15] and Recurrent visual attention model[3] used to solve the problem of hierarchical image classification on histology datasets. Method [27] is available online on github and because of time constrains has not been re-implemented and only used for comparison in experiments.

A thorough explanation of each step in the implementation will be given as well as proposed modifications to the original ideas given by the authors of the methods.

4.2 Recurrent visual attention model

As proposed by [3] the recurrent visual attention model (shown in 4.1) consists of three networks with parameters θ_a, θ_l and $\theta_x$providing appearance and location. Utilizing a recurrent hidden state h_t, we aggregate information from a sequence of glimpses to classify tissue images. We are going use the same framework parameterized as $\theta = \{\theta_a, \theta_l, \theta_x\}$, but parts of the internal networks will be changed to optimize the classification accuracy.

We will also try to follow the same experiments. In [3], the framework was tested on the CAMELYON16 dataset with a dataset preprocess, which will be used in the same manner.

4.2.1 Framework

First, we are going to modify the structure of the already existing algorithm [3].

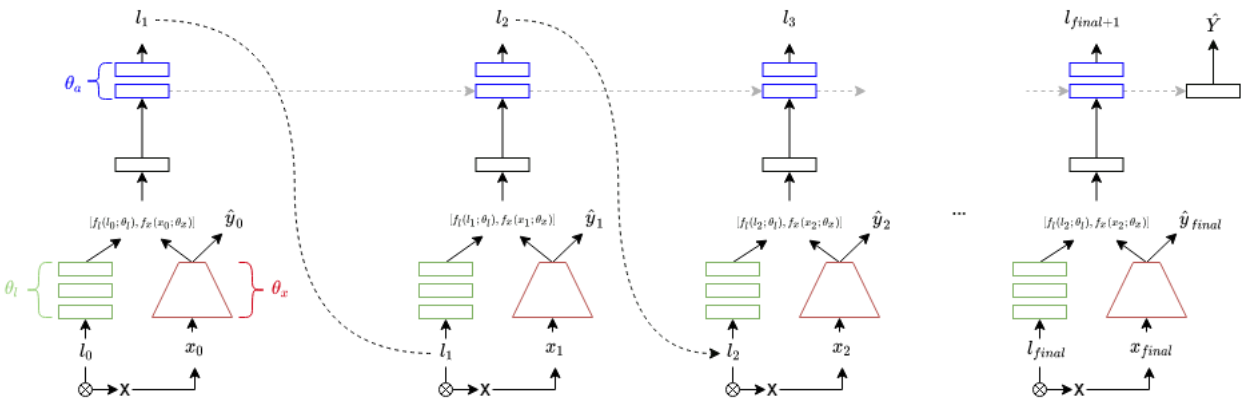


Figure 4.1: Modified structure of the proposed Recurrent visual attention model [3]. Hadamard product is no longer present in the feature combination. Instead a concatenation is done.

Then, let us go step by step through the components of the model

Spatial network

A patch x_t is extracted from the slide tile at a location l_t, by applying an affine transformation on the input image.

I parameterized the location as $l_t = \{\delta_x, \delta_y, \mu_x, \mu_y\}$, where δ is the scale of the glimpse and μ is the center location of the glimpse along each axis. In an affine transformation setting, we use the location parameters to create an affine transformation matrix T

$$T = \begin{bmatrix} \delta_x & 0 & \mu_x \\ 0 & \delta_y & \mu_y \end{bmatrix} \tag{4.1}$$

where μ_x and μ_y represent the translation of the transformation and δ_x and δ_y the scale of the transformation. Other parameters are always 0, because those introduce the shearing of the image, which would unnecessarily stretch the transformed patches. This affine matrix is passed to an affine grid generator function torch.nn.functional.affine_grid(theta, size) (provided by PyTorch[20]), where *theta* is our affine matrix and $size = (H, C, H_{out}, W_{out})$ determines the dimensions of the output grid. This grid is then passed to a sampling function torch.nn.functional.grid_sample(input, grid)[20] along with the input, where $input \in \mathbb{R}^{N \times C \times H_{in} \times W_{in}}$ and $grid \in \mathbb{R}^{N \times H_{out} \times W_{out} \times 2}$. The sampler samples the input according to the given affine grid to produce a patch of target size 304×304 pixels.

The first location i. e. the first matrix is always set to an identity matrix to extract only a coarse representation of the entire tile as said in [3].

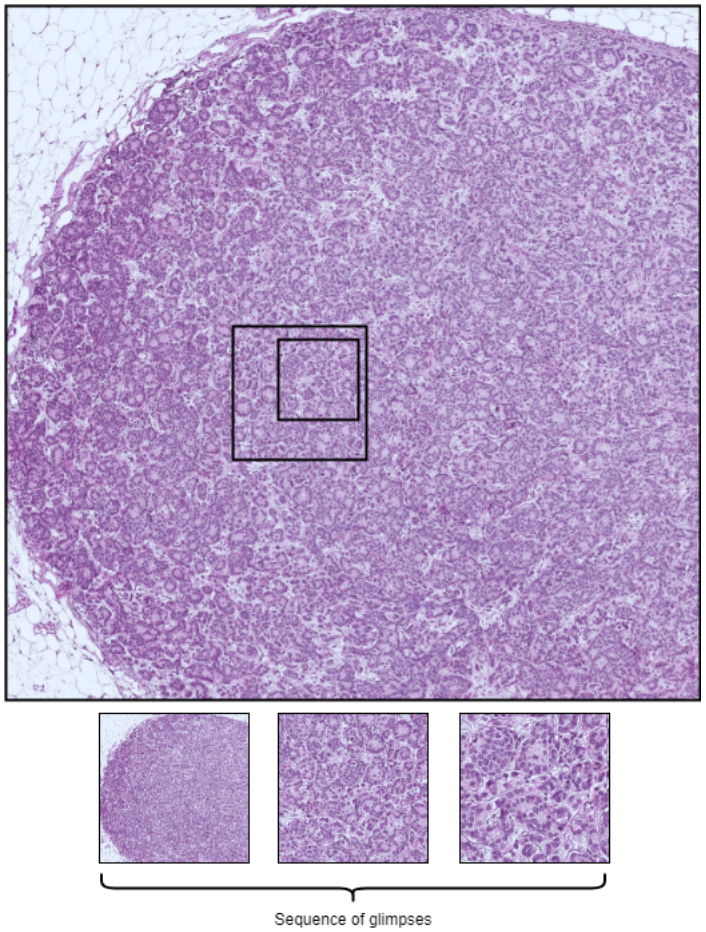


Figure 4.2: Extracted sequence of glimpses at different locations and scales. The sequence starts with a coarse representation of the entire image.

The appearance network θ_x then takes in the extracted and resized patch to create the feature representation of a given patch. The network is essentially a CNN network, which they [3] proposed. I chose the network to be based around a patch-based deep convolutional neural network [13], where it seemed to have good results (around 95% accuracy) in patch classification. This particular CNN was used by [13] to train the model on patches extracted from entire WSIs of The Cancer Genome Atlas Program dataset. The structure is present next:

Layer	Filter size, stride	Output $W \times H \times N$
Input	-	$400 \times 400 \times 3$
Conv	10×10, 2	$196 \times 196 \times 80$
ReLU+batchNorm	-	$196 \times 196 \times 80$
Max-pool	6×6, 4	$49 \times 49 \times 80$
Conv	5×5, 1	$45 \times 45 \times 120$
ReLU+batchNorm	-	$45 \times 45 \times 120$
Max-poll	3×3, 2	$22 \times 22 \times 120$
Conv	3×3, 1	$20 \times 20 \times 160$
ReLU	-	$20 \times 20 \times 160$
Conv	3×3, 1	$18 \times 18 \times 200$
ReLU	-	$18 \times 18 \times 200$
Max-pool	3×3, 2	$9 \times 9 \times 200$
FC	-	320
ReLU+Drop	-	320
FC	-	320
ReLU+Drop	-	320
FC	-	256 (Configuration dependant)
Softmax	-	256 (Configuration dependant)

Table 4.1: Modified model architecture of a patch-based CNN for slide classification[13]. ReLU+batchNorm is a sequence of Rectified Linear Units (ReLU) followed by a batch normalisation layer. ReLU+Drop is a sequence of ReLU followed by a dropout layer with a propability of 0.5. The size of the output is the size of the hidden layer of the glimpse + the hidden layer of the location.

The output of this network will be the appearance features for the current patch x_t: $f_x(x_t; \theta_x)$

Spatial and location combination

Using the location network θ_l, which is fed the current location l_t in timestep t I got the location representation by using 3 fully connected layers: $W_1 \in \mathbb{R}^{4\times32}$, $W_2 \in \mathbb{R}^{32\times128}$ and $W_3 \in \mathbb{R}^{128\times256}$. This gives us our location features $f_l(l_t;\theta_l)$.

Having the appearance and location features, we can combine them using a simple concatenation into a tensor $\in \mathbb{R}^{N\times256+256}$, where N is the batch size. This concatenated tensor is then passed through a sigmoid activation layer. This gives us the glimpse representation g_t.

$$g_t = ReLU\Big(\Big[W_3\Big(ReLU\Big(W_2\big(ReLU(W_1(l_t))\big)\Big)\Big), CNN(x_t)\Big]\Big) \tag{4.2}$$

where [] represents concatenation.

Attention network

As defined in [3] the attention network θ_a aggregates glimpse representations g_t into our hidden state of the network h_t and based on the resulting h_{t+1} outputs the next location parameters l_{t+1}.

I decided to construct the network with the following notion. The hidden state from the previous timestep t is fed through a fully connected layer $h_{fc} \in \mathbb{R}^{512\times256}$. Similarly the current glimpse representation g_t fed through a different fc layer $g_{fc} \in \mathbb{R}^{512\times256}$. The state h_{t+1} is then obtained by concatenating the features from g_{fc} and h_{fc} followed by a ReLU unit.

$$h_{t+1} = ReLU([h_{fc}(h_t), gfc(g_t)]) \tag{4.3}$$

Given the new hidden state h_{t+1} the network can now output predictions for the next location. I used the reparametrisation trick [18] to sample the next location from parameterized probability distributions. This is done by

passing the new h_{t+1} through a fc layer $featFC \in \mathbb{R}^{512\times128}$ and then the features through a final fc layer $featMU \in \mathbb{R}^{128\times4}$ and a hyperbolic tangent function, which outputs μ_l for every location parameter δ and μ. We then create normal distributions parameterized by μ_l and a standard deviation, which I chose to be 0.05 at all times. The next location parameters are sampled from these distributions to find the next location l_{t+1}. This new location now needs to be clamped between -1 and 1, since the affine matrix, to which l_{t+1} is the input, uses only variables in the range $[-1, 1]$.

After all glimpses have been extracted and aggregated, we are going to pass the final hidden state $h_t|[x_1..x_t]$, which has all the glimpses $g_{1..t}$ into a fully connected layer $S \in \mathbb{R}^{128\times2}$ and the output should be the predictions for the slide label $Q(Y^{(i)}|x^{(i)}_{[i:T]})$ where (i) denotes the batch number and T the number of glimpses

4.2.2 Training

I have done the training of the algorithm mostly in the same way as did the authors.

Then I used a Cross entropy function:

$$L_c = -\sum_i Q(Y^{(i)}|x^{(i)}_{[i:T]}) \log(Y_i) \tag{4.4}$$

where Y_i are the targets for batch i, in this case image-level labels. I used it to output the image-level loss L_c and the patch-level loss L_p. The image-level loss is computed using the slide predictions $Q(Y^{(i)}|x^{(i)}_{[i:T}$ and the patch-level loss by applying a fully connected layer $f_{patch} \in \mathbb{R}^{256\times2}$ on the glimpse spatial features f_x.

$$L_p = -\sum_i \sum_t f_{patch}(f_x(x_t;\theta_x))_i \log(Y_{i,t}) \tag{4.5}$$

where t denotes the glimpse number or timestep and i is the batch number. We obtain $Y_{i,t}$ by repeating each batch label Y_i t times. The final objective

loss, which the algorithm minimizes, is then the same as in [3], but stripped of two losses L_a and L_l, because they did not help guide the network to better results.

$$L = L_c + L_p \tag{4.6}$$

Training was also done on my implementation of the original proposed framework [3]. The difference in the implementation being that the glimpse representation is not a concatenation, but a piece-wise multiplication and the glimpses are not concatenated in the hidden state, but summed up. The training results of this original implementation were not satisfactory, so I opted for adding a reinforcement loss L_r, which derives from this article [8]. To obtain this loss, we need to add another FC layer $W_r \in \mathbb{R}^{128\times 1}$ to the network after we get the new h_t from the attention network. We pass the h_t through this layer to b_t, which regresses the baseline in the reward function to reduce the variance of the gradient update.

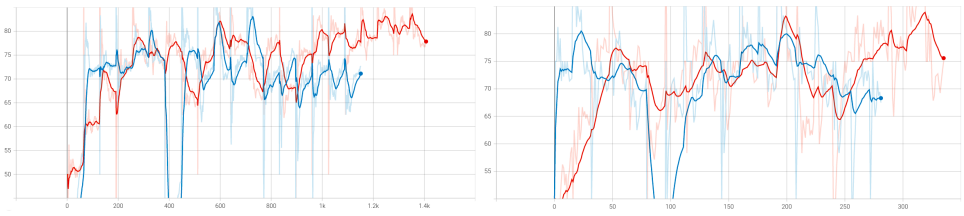
$$R = repeat \quad t \quad times(\hat{Y}^i == Y^i) \tag{4.7}$$

The reward function is a comparison between the ground truth of the image and predicted label $\hat{Y}^i$ for each batch i. Each result is then repeated t times for the number of glimpses.

$$L_r = \sum_{i=0}^{N}\sum_{t=0}^{T}(-lp_{i,t}(R_{i,t} - b_{i,t})) \tag{4.8}$$

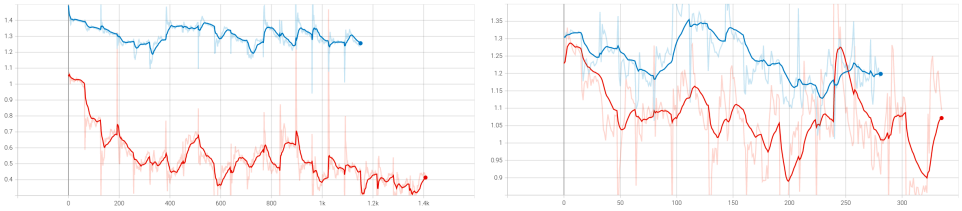
where $lp_{i,t}$ is the propability density function (PDF) computed from from the normal distributions described in the attention network.

Using the modified model and the original implementation with the add reinforcement learning loss, I managed to get the following training results:



(a) : Training acc. **(b) :** Validation acc.

Figure 4.3: Accuracy of classification during training. Training for the original method with L_r (Red) converging at around 85% acc and validation at around 84% acc. While the modified method (Blue) converges at around 75% and the validation set at around 72%. The x axis corresponds to $batchnumber \times epochs$



(a) : Training loss. **(b) :** Validation loss.

Figure 4.4: Loss progression through training. The x axis corresponds to $batchnumber \times epochs$. Original with L_r (red) and modified method(blue)

Other modifications

I also tried using the retina-like patch extraction proposed in [8]. This replaces our affine matrix transformation with retina-like mechanism. This mechanism extracts a sequence of scaled down images at a location l_t at a timestep t. The sequence is concatenated, which produces the following feature: the center part of the image is kept at a high resolution, while the resolution is progressively lowered for pixels further from the center location. l_t, apart from the affine matrix, is defined as $l_t = \{\mu_x, \mu_y\}$, where μ are the center coordinates of each axis of the sequence. Since l_t is parametrised only by the center coordinates, the scale, which was δ in the previous section, is a function of a position that is non-trainable. The sequence of patches is then concatenated into one tensor. This gives the most attention to the part of the image where the most scaled down patch is.

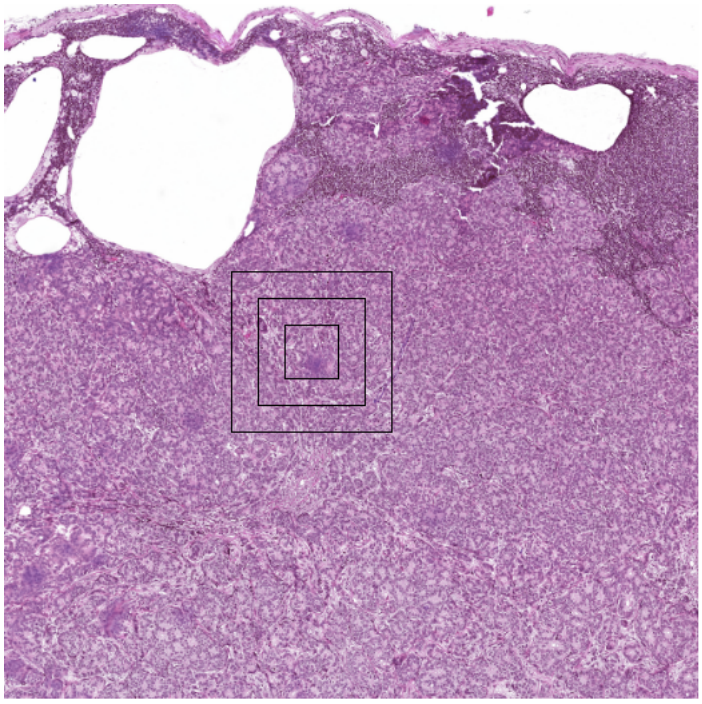
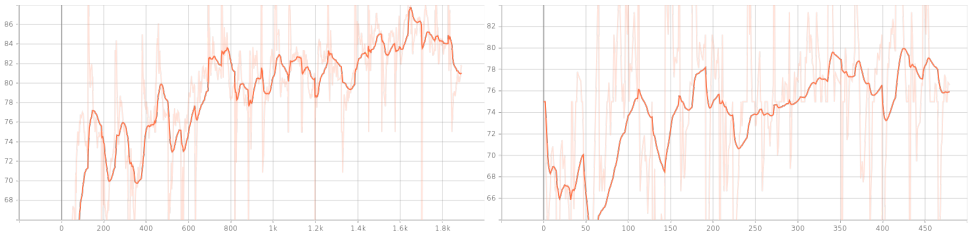


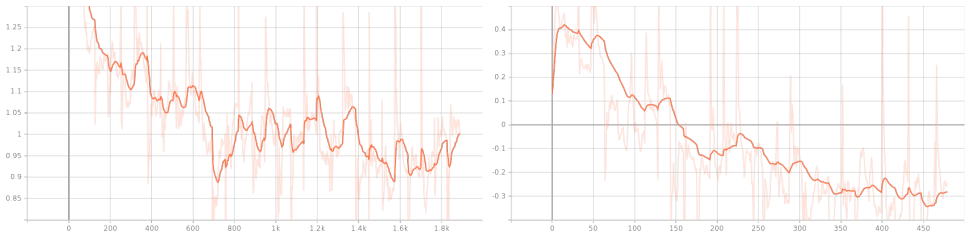
Figure 4.5: Retina extraction of a sequence comprising of 3 patches at a location l_t

I tried the same training as in the previous section with this model component:



(a) : Training acc. **(b) :** Validation acc.

Figure 4.6: Accuracy of classification during training. Training converging at around 86% acc and validation at around 82% acc. The x axis corresponds to *batchnumber* $\times$ *epochs*



(a) : Training loss. **(b) :** Validation loss.

Figure 4.7: Loss progression through training. The x axis corresponds to *batchnumber* × *epochs*

This component yielded better results than the affine matrix extraction modification, but also had worse validation results than the original method with the added L_r loss.

4.3 Quadtree approach

We implement the Quadtree hierarchical approach described in Section 3.2 and in [15]. First, we are going to implement the quadtree segmentation.

4.3.1 Quadtree segmentation

The goal of this subsection is to implement an algorithm, which creates a quadtree structure from a given image and in this sense splits the image into smaller patches of the same size. I implemented the algorithm in the following way.

Algorithm 2: Quadtree decompostion

```
Struct Node contains
    children = list[4]
    size = Int
    coords = [h_x, h_y, w_x, w_y]          ▷ Bounding rectangle of a region

def Subdivide:
Input  : Node: N, threshold: k, minimum pixel size: p, criterion c,
         original_Image : i ∈ R^{H×W×3}
Output : Node: N
:
if c {N, i}≤ k   or   N.size≤ p then
    return
else
        ▷ Split rectangle into 4 rectangles of the same size
    split N.coords into 4 subquadrants of equal size[cords1, cords2, cords3, cords4]

    for cord in [cords1, cords2, cords3, cords4] do
        newNode = Node(cord)
        newNode.size = getSizeFromCoords(cord)
        Subdivide(newNode, k, p, c, i)
        N.children.append(newNode)
```

As described in [15] a root node N_{root} is created, which holds the coordinates of the entire input image and the size of the shortest edge of the bounding box. In our case, the initial coordinates are $[0, 0, 5000, 5000]$ describing a bounding box around the entire image and initial size is 5000. The N_{root} is then given to the Quadtree decomposition algorithm along with a minimum pixel size k, which determines the minimum size of the patches we want to segment, criterion c, which computes a mean value out of the quadtree region (criterion is closely described in 4.3.1) that is compared with the splitting threshold k, to determine whether the algorithm should split further. Finally, we pass the original image i into the algorithm , to extract a subregion around specified coordinates. This subregion serves as an input to the c function.

Next, we are going to be optimizing the splitting threshold and creating a criterion, which will split the incoming image into favourable regions. Favourable regions being background, which will be split at a low resolution, because it is uninformative and foreground (tissue), which we want to be split into patches at the highest resolution possible, since it is the informative part.

Choosing the right criterion

In [15] the best performing models were ones using the Heamatoxylin criterion. In the article it was done by using color deconvolution to separate out the heamatoxylin channel. The mean of this channel was then the output of this criterion. I will be implementing this criterion and using it in the segmentation.

The dataset we are working with has been stained by the H&E (haematoxylin and eosin) compounds. I used **colour deconvolution** [22] to separate haematoxylin and eosin as colour channels using the histomicsTK [16] library. The deconvolution algorithm works like this: Let us annotate an input RGB image as $I_C \in \mathbb{R}^{M \times N \times 3}$, where $\{M, N\}$ represents the image dimensions and C corresponds to the color channels. Then the light before entering the specimen is defined as $I_{O,C} \in \mathbb{R}^{M \times N \times 3}$, which is in practice initialized to 256 on all color channels (white light). An RGB image used in the deconvolution is then formally defined as

$$I_C = I_{O,C} \exp(-OD_C) \tag{4.9}$$

using Labert-Beer's law. From above definitions an optical density image $OD_C \in \mathbb{R}^{M \times N \times 3}$ is constructed by

$$OD_C = -\log_{10}(I_C / I_{O,C}) \tag{4.10}$$

now lets define an optical density matrix matrix $ODM \in \mathbb{R}^{3 \times 3}$, where each row of this corresponds to a stain compound heamatoxylin, eosin or DAB. I chose to initialize this matrix as

$$ODM = \begin{bmatrix} 0.65 & 0.70 & 0.29 \\ 0.07 & 0.99 & 0.11 \\ 0.00 & 0.00 & 0.00 \end{bmatrix} \tag{4.11}$$

according to [22]. The last row is zeroes, because it corresponds to the DAB compound, which is not present in our dataset.

ODM is then normalized and an inverse of this normalized matrix is computed resulting in a matrix D

$$D = (norm(ODM))^{-1} \tag{4.12}$$

For the ODM to be invertible, we first need to complement the third row by a normalized cross-product of the first two rows. This is then added in place of the zeroed row. Image is then deconvoluted by applying the dot product on OD_C with D. This deconvoluted image is tranformed back to RGB space by applying the inverse of (4.10). Individual channels are different stains in the case of this RGB deconvoluted image. The deconvolution is shown below.

$$Image_{deconv} = -\log_{10}(I_C/I_{O,C}) \cdot (norm(ODM))^{-1} \tag{4.13}$$

With an example of tile deconvolution.

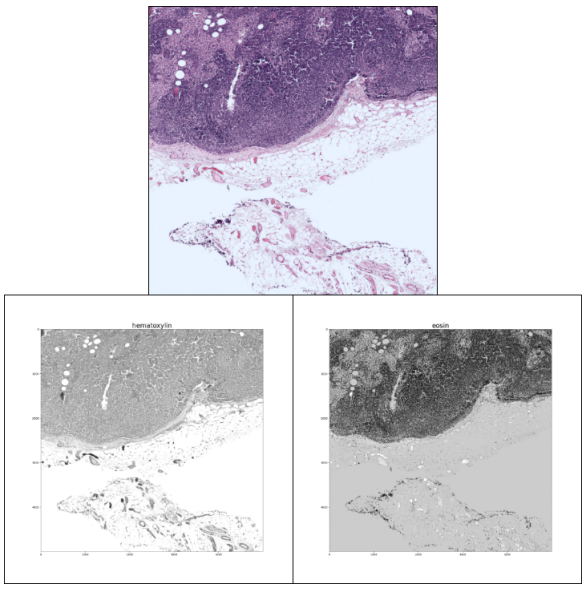


Figure 4.8: Example of color deconvolution applied to a tissue tile. Original tile on top, Heamatoxylin channel on the left and eosin on the right

Threshold

I computed the threshold as described in [15] by applying the criterion 4.3.1 on all image tiles. From all criterion outputs, the mean μ and standard deviation σ was computed resulting in the following values by using deconvolution matrix (4.11)

variable	value
μ	4389.97
σ	737.34

The resulting threshold choice is then calculated from

$$threshold = \mu - \sigma \tag{4.14}$$

Here I chose -1 as a hyperparameter, since that yielded the best results in [15] resulting in a threshold value of $\tilde{3}652.627$.

Using this criterion and this threshold value, the Quadtree extraction yielded the following results.

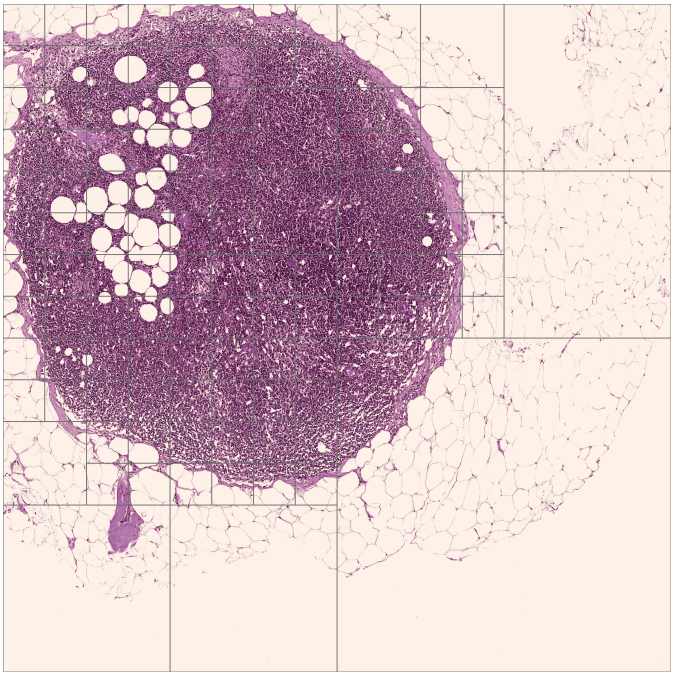


Figure 4.9: The results of the Quadtree segmentation on a tissue tile

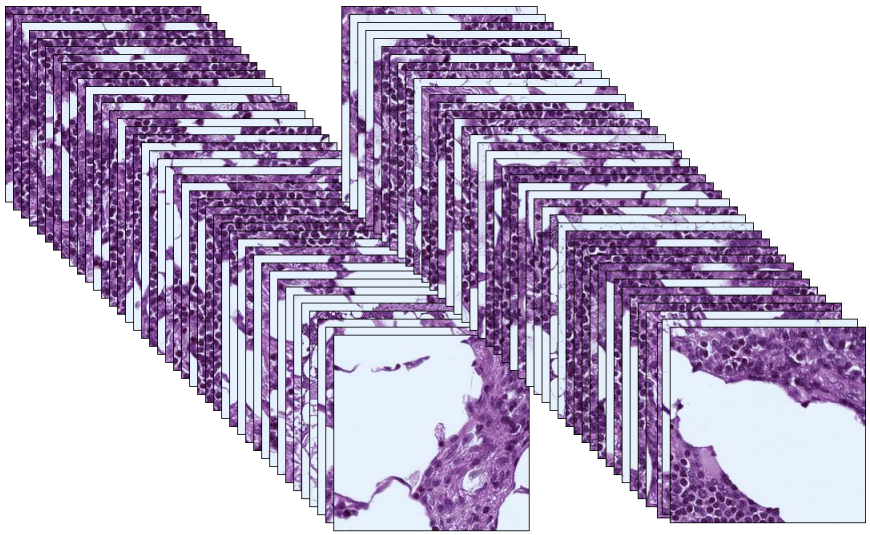


Figure 4.10: Segmented patches from a tissue tile

4.3.2 Network architecture

I split the model architecture into two parts as described by [15]. First, we are going to extract features for every leaf node image x_p, which has been segmented by the Quadtree algorithm. All x_p have been resized to size 256×256 pixels. After this, we are going to feed the features into CLAM [27] classification network, which utilizes the clustering mechanism and classify each image according to the pathes x_p.

Feature extraction

For the feature extraction part of our model, I chose ResNet50 [26] to be my main feature extractor and used it as a non-trainable part of the network. I did not use the default implementation by Pytorch, but instead opted for a modified network proposed by [27]. This network is comprised of three convolution blocks making use of the skip connections from the original implementation and adding a mean-spatial-pooling layer after the last block to output 1024-dimensional feature vectors. I initialized the weights of the model as pretrained on the ImageNet[7] dataset.

Classification

The 1024-dimensional features for each patch x_p are sent with the label assigned to the image from which the features were extracted, to the CLAM 3.3 network. I used the predictions from the MIL2.5 baseline, as well as the clustering algorithm to train the network.

4.3.3 Training

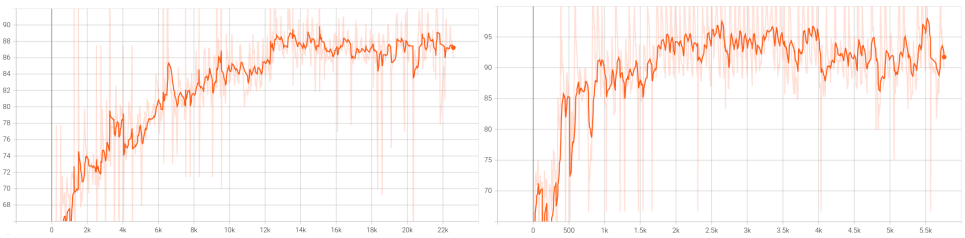
I trained the model by minimizing an **objective function** comprised of two smooth top-1 svm [4] loss functions.

$$l(s, y) = \max\left\{\left(\max_{j \in Y \setminus \{y\}} \{s_j + 1\} - s_y\right), 0\right\} \tag{4.15}$$

where s are predictions for each label and y is the ground truth label (bag label or instance label). The loss is zero if the ground truth score is higher than all other by at least 1. Otherwise it produces a penalty, which is the difference between the ground truth prediction and the highest score of the other classes (in our binary classification case there are no other classes, there is only a second class). The final summed loss L is obtained by

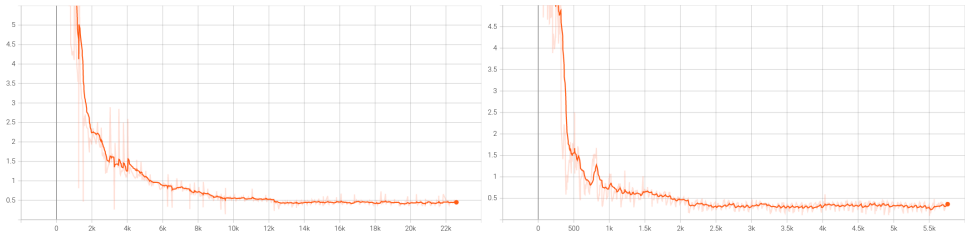
$$L = BL + CL \tag{4.16}$$

The bag loss BL is given by applying the loss function to the output of the MIL attention layers and the clustering loss CL by applying the loss function to the output of the attention layers of the clustering algorithm. Entire training progress can be seen below, where the model has been applied on patches extracted from the Quadtree segmentation.



(a) : Training acc. **(b) :** Validation acc.

Figure 4.11: Accuracy of classification during training. Training converging at around 87% acc and validation at around 93% acc. The x axis corresponds to *batchnumber* × *epochs*



(a) : Training loss. **(b) :** Validation loss.

Figure 4.12: Loss progression through training. The x axis corresponds to *batchnumber* × *epochs*

From these figures, we can see that the training set converged with its accuracy at around 87% and the validation set at around 93%. This is an interesting behaviour, because we would expect the training accuracy to be higher than the validation. It is impossible for the training and validation set to get mixed up, so the training results are valid. Only possible reason would be that the validation set had a bias towards one label, which is unlikely, since the data was split proportionally to each other having the same ratio of negative to positive labels. The losses of both sets steadily decreased over epochs and correspond with the accuracy presented.

Chapter 5

Data description

Today, classification methods can be tested on different datasets, like for example The Cancer Genome Atlas (TCGA)[1] [9], which provides a large database of cancer related data from different parts of the body. Other widely used datasets being the CAMELYON16 [2] and CAMELYON17[2] [5], which are datasets of sentinel lymph nodes of breast cancer patients. In this work, I tested all methods on the CAMELYON16 dataset.

5.0.1 CAMELYON16

This dataset is part of a grand challenge[3], which was held in 2016. The goal of this challenge was to evaluate new and existing algorithms for breast cancer classification. The data was collected from two different medical centers. Radboud University Medical Center (Nijmegen, the Netherlands) and the University Medical Center Utrecht (Utrecht, the Netherlands) contributed to create a training and testing dataset.

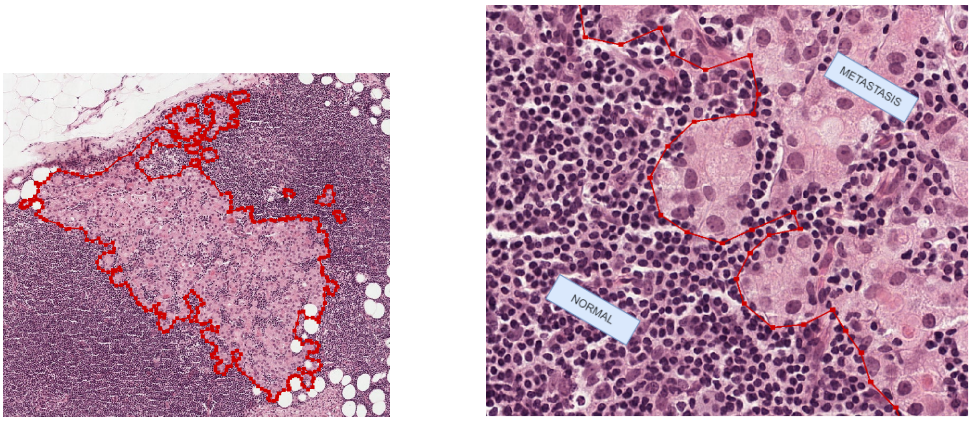
[1] https://www.cancer.gov/about-nci/organization/ccg/research/structural-genomics/tcga

[2] https://camelyon17.grand-challenge.org/

[3] https://camelyon16.grand-challenge.org/Home/

Center	Train	Test	Normal	Metastasis
RUMC	170	79	150	99
UMCU	100	50	90	60
Total	270	129	240	159

This dataset, apart from its WSIs, contains slide-level annotations and pixel-level annotations, which provide a region in the tumor images, where exactly the tumorous tissue is.



(a) : Annotated tissue region **(b) :** Magnified tissue region

Figure 5.1: Annotated tissue slide, which was produced by the ASAP tool [24]

5.0.2 CAMELYON16 tiles

I prepared a dataset of tiles extracted from tissue slides using the Pyhist [17] library. Each tile was extracted at a magnification level of 2, which resulted in 5000×5000 pixels large tissue tiles.

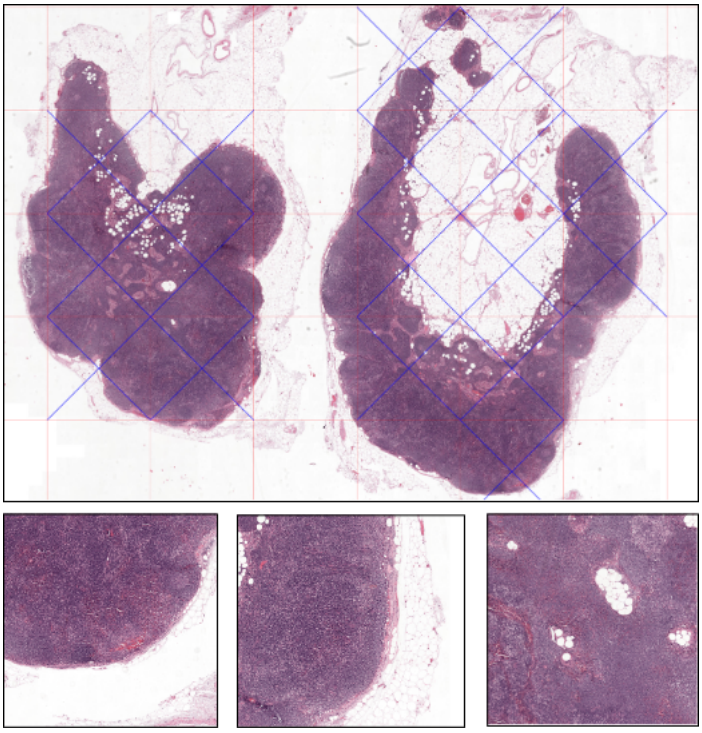


Figure 5.2: Tissue slide and some extracted tiles. The crosses on the WSI represent the picked regions, while the red lines represent the grid, from which tiles are selected.

The tiles were than provided to two of the methods (Quadtree and Recurrent visual attention), while I provided entire WSIs to the CLAM method, because it classifies images based on 256×256 large patches and has its own preprocessing algorithm.

Chapter 6

Results

In this chapter, I will compare the results of the methods I implemented (Quadtree approach and Recurrent visual attention) as well as show the differences in performance from state-of-the-art methods like CLAM [27]. I have used the CAMELYON16 dataset for all of the testing. All tests were done on GTX 1080 Ti GPUs, where in the case of Quadtree and CLAM testing multiple GPUs were provided.

I decided to test the methods on how well they can classify each image tile, which has been annotated either normal or metastasis (0 or 1) and then try testing them by seeing how well they can classify entire WSIs according to aggregated results from extracted tiles.

6.1 Tiles

Each of the two methods, which I implemented were tested on never seen before tissue tiles that have been labeled according to the annotations provided in the CAMELYON16 dataset. The number of tiles was approximately 330. I will first show the settings on which the methods have been trained on and then tested. Only the best performing models on the validation data were selected. The original RVAM with the added L_r loss and the Quadtree method (which used the deconvolusion criterion).

name	value	name	value
epochs	57	initial lr	2e-4
batch size	1	patch size	244×244
CLAM used	yes	weight decay	1e-5
threshold	3283.93	initial image size	5000×5000
train/val split	0.8/0.2	lr_reduction	after 20 epochs

Table 6.1: Configuration for the Quadtree approach

name	value	name	value
epochs	27	initial lr	1e-4
batch size	4	patch size	304×304
glimpses per tile	12	initial image size	5000×5000
train/val split	0.8/0.2	lr_reduction	after 5 epochs

Table 6.2: Configuration for the Recurrent visual attention model

lr_reduction means that the model reduced its learning rate after it has not improved for a specified set of epochs. Besides accuracy, I also computed the **Area Under Curve (AUC)** score for each testing set. AUC is the measure of the ability for the classifier to distinguish between classes. The bigger the AUC score, the better the model is able to correctly predict a class (0 AUC meaning the classifier predicts everything wrong and 1 meaning the classifier is able to predict every class correctly). If AUC score is in the range 0.5<AUC<1, then it is more likely to predict true positives and true negatives than false positives and false negatives. The results on the tile-level classification follow. Here 'ref' refers to reference.

Method	Accuracy	AUC
RVAM	87%	0.89
RVAM ref(L_p + L_c)	86%	0.84
RVAM ref(L_p + L_c + L_a)	96%	0.95
Quadtree	92%	0.84
Quadtree ref	95%	0.97

I got a difference in accuracy of 3% compared to the reference Quadtree results [15], but the AUC was lower by a much bigger margin. I managed to get almost the same accuracy with my RVAM compared to the RVAM reference which used only the L_p and L_c losses, whereas the reference did much better with three losses adding the L_a loss [3]. I was unable to achieve

better results by adding the attention L_a loss, where its implementation resulted only in worse performance.

Here, I plotted the Receiver Operator Characteristic (ROC) curve, which is an evaluation metric for binary classification problems. It is a probability curve that plots the true positive rate against false positive rate at various threshold values and essentially separates the 'signal' from the 'noise'.

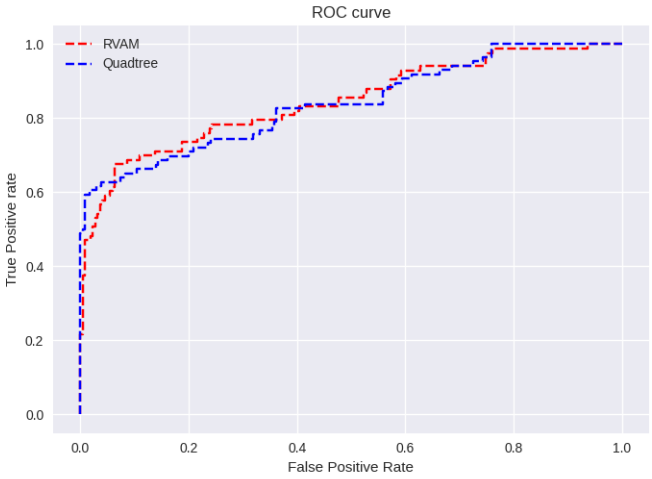


Figure 6.1: ROC curve for Recurrent visal attention model (RVAM) and Quadtree

Overall, the ROC curve of the Recurrent visual attention model is higher than the Quadtree curve. Therefore we can asses that the Recurrent visual attention model performed better in classifying the positive class (metastasis) in the dataset even if it had worse accuracy than the Quadtree.

6.2 Slides

I then tested the trained models on testing slides. I did this by evaluating tiles for every slide and then aggregating the probabilities by creating a sum for all tiles to form a prediction of the slide label. I also trained the CLAM [27] model on the CAMELYON16 slides and then ran the algorithm 3.3 to test it. The CLAM model is only in slide testing, since it does not train on tiles, but rather extracted patches and has its own algorithm, which I did not implement, but only used. It is made public and available through [27]. Here

are the settings which I chose for the CLAM training and testing.

name	value	name	value
epochs	50	initial lr	2e-4
B	8	patch size	256×256
train/test split	0.75/0.25	weight decay	1e-5

Table 6.3: Configuration for CLAM model

where B is the number of positive/negative patches to sample for the clustering mechanism 3.3. The results of the testing follow. All were done on approximately 50 testing slides.

Method	Accuracy	AUC
RVAM	76%	0.56
RVAM ref(L_p + L_c)	84%	0.84
RVAM ref(L_p + L_c + L_a)	93%	0.95
Quadtree	81%	0.60
CLAM	99%	0.99
CLAM ref	-	0.95

My models achieved lower accuracy in the slide-level classification than the reference, but still the Quadtree approach performed better than RVAM by approximately 5%. The CLAM testing which, I did by using their algorithm, outperformed all other models by a lot, having accuracy approximately of 99% and an AUC score of approximately 0.99. The reference [27] gives us only an AUC score for comparison, which was tested on a combination of CAMELYON16 and CAMELYON17 dataset, so we are unable to make a full comparison, since my CLAM testing was done only on the CAMELYON16 data.

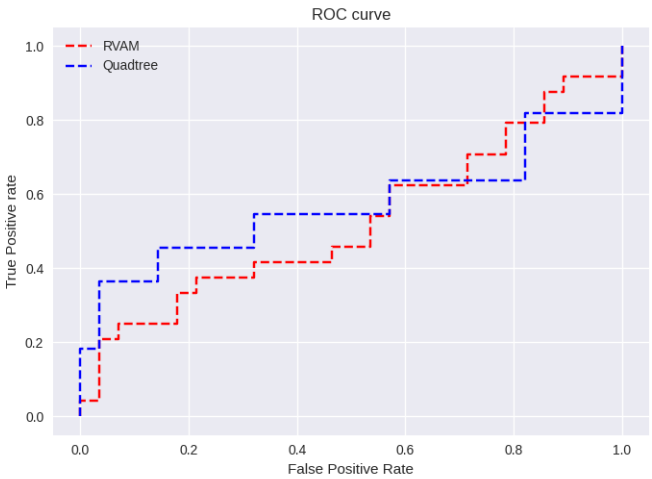


Figure 6.2: ROC curve for slide classification of Recurrent visual attention model (RVAM) and Quadtree

The ROC curve of the slide classification is worse in comparison with the tile-level ROC curve. Overall, the Area Under the ROC curve (AUC-ROC) is better for the Quadtree implementation, which means that it classified positive cases better than RVAM overall.

Chapter 7

Discussion

In this chapter we perform final analysis of additional information, which differentiated some methods and may prove useful when deciding on which method to choose for histology image classification.

Firstly, the size of data storage restrictions may be the deciding factor, when one is unable to store large quantities of data onto a machine. The CLAM model, even if the best performing, is very demanding on storage capacity. When using the framework, the entire space occupied by extracted patches and feature vectors was around 98GB. On the other hand, the Quadtree approach, which exhaustively extracts only the foreground, needed around 15GB of total storage for the patches, which is much less than in the CLAM case. Finally, the Recurrent visual attention network does not save any files prior to its classification mechanism and uses only the extracted tiles.

Another issue is the specifications of the tiles, which are used in the Quadtree and RVAM methods. Although the Quadtrees are able to extract patches from tiles larger than 5000×5000 pixels, the RVAM method is unable to sample the patch location from images bigger than this. In my experiments, I tried using tiles of size 8192×8192 pixels, but it was computationally impossible, which was also proved in the reference [3].

The time to train a model varied from method to method. While CLAM and Quadtree methods took a long time to prepare the dataset for its classification (patch and feature extraction in the case of CLAM and patch extraction in the case of Quadtree) sometimes taking over 24 hours to complete, the training was then quick. Quicker in the case of the Quadtree (approximately

3 hours), since it had less patches to load. On the other hand RVAM training took much longer (approximately 10 hours) because it has to deal with entire tiles.

When implementing the RVAM method, the loss L_l is proposed in the reference, which should encourage the exploration of the network. Although this loss seems like it was described, but not used in experiments, I also tried implementing it in my version of the model. It did not improve the classification of the images, but rather led the network to explore areas furthest from each other leading to an extraction of images which were unusable for correct classification.

What surprised me was that the Quadtree method was not used for WSI image classification in the reference, but only for tile classification. In this notion I was unable to compare the slide-classification results from my implementation with the reference, which begs the question if the Quadtree method is even suitable for WSI classification. The results tell that it is able to partially distinguish between tumorous and normal slides, but is still worse in comparison to the state-of-the-art methods.

Chapter 8

Conclusion

In this thesis, we compared some existing methods for hierarchical histology image classification, mainly the Quadtree method [15], Recurrent Visual Attention Model (RVAM) [3] and the Clustering-constrained Attention Multiple instance learning mechanism (CLAM) [27]. We achieved this by implementing the RVAM and Quadtree approaches according to the respective articles and used the already existing CLAM implementation.

We used the CAMELYON16 dataset to evaluate the training results and made modifications for improvement. We were unable to implement a successful RVAM model by following instructions from the article, so the original model was modified to provide better results by introducing a reinforcement learning loss and a CNN as a part of the appearance network. The Quadtree network was not modified, but successfully re-implemented according to the authors description of the approach. We gave proof that the models are able to train on training tiles extracted from WSIs and give satisfactory results on validation data.

Different approaches to tile/patch extraction were utilized for data preprocessing, namely the patch extraction technique used in CLAM and the Quadtree patch extraction.

The trained models from each method were tested on tiles and slides from the CAMELYON16 dataset. We compared all methods with each other and with their references by their accuracy, Area Under Curve (AUC) score and Receiver Operator Characteristic (ROC) curve. Overall the methods yield good results on the tile testing when compared with its references, but rather

unsatisfactory results on the slide testing. The best performing method, which I implemented was the Quadtree and overall the best results were obtained from the CLAM. This might be due to a badly chosen aggregator function, but because of time-constrains, alternative experiments could not be done. Finally, other theoretical comparisons were discussed, which could guide the reader to choose the correct method for their task. The best performing method was the CLAM, which provided almost 99% classification accuracy on WSI classification, but it was very time-consuming to preprocess the data, so an alternative would be the Quadtree method, which substantially reduces the number of patches needed for training. The best method to choose for tile classification would be the Quadtree, where it provided around 92% accuracy. While the RVAM method performed the worst, it was also the quickest way to train a model and obtain around 87% accuracy on tile classification.

Both the Quadtree and the CLAM method could be used in a sub-typing setting, where more than 2 labels are present, since they use the clustering mechanism. The best choice from these two would be the CLAM method, because it was originally designed to classify cancer sub-types. Quadtrees were not tested on this task, so further experiments would have to be made to make a valid comparison between the two approaches. The RVAM was not designed for this problem, but could also be used here if the sub-typing task was reduced to a binary classification.

There are still improvements to be made if one were to continue from my implementations. Mainly the slide classification could be improved by devising a better solution for the aggregation of tile predictions. An alternative location representation and new location extraction could be proposed to the RVAM to guide the network to better attend to discriminate areas.

Bibliography

[1] Global Transformational Health Research Team at Frost & Sullivan. Global tissue diagnostics market, forecast to 2022. *Frost and Sullivan*, 2018.

[2] B. E. Bejnordi, M. Veta, P. J. Van Diest, B. Van Ginneken, N. Karssemeijer, G. Litjens, J. A. Van Der Laak, and et. al. Diagnostic assessment of deep learning algorithms for detection of lymph node metastases in women with breast cancer. *JAMA - Journal of the American Medical Association*, 318:2199–2210, 2017.

[3] BenTaieb, Aïcha, , and Ghassan Hamarneh. Predicting cancer with a recurrent visual attention model for histopathology images. *In Proc. of MICCAI 2018*, pages 127–137, 2018.

[4] Leonard Berrada, Andrew Zisserman, and M Pawan Kumar. Smooth loss functions for deep top-k classification. *International Conference on Learning Representations*, 2018.

[5] P. Bándi, O. Geessink, Q. Manson, M. Van Dijk, M. Balkenhol, M. Hermsen, B. Ehteshami Bejnordi, B. Lee, and et. al. From detection of individual metastases to classification of lymph node status at the patient level: The camelyon17 challenge. *IEEE Transactions on Medical Imaging*, 38:550–560, 2019.

[6] de Haan, K., Zhang, Y., Zuckerman, and J.E. et al. Deep learning-based transformation of h&e stained tissues into special stains. *Nat Commun*, 8 2021.

[7] 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, pages 248–255. Ieee, 2009.

[8] Volodymyr Mnih et al. Recurrent models of visual attention. *NIPS*, page 2204–2212, 2014.

[9] Deng F, Shen L, Wang H, and Zhang L. Classify multicategory outcome in patients with lung adenocarcinoma using clinical, transcriptomic and clinico-transcriptomic data: machine learning versus multinomial models. *Am J Cancer Res.*, 12:4624–4639, 2020.

[10] Adam Goode, Benjamin Gilbert, Jan Harkes, Drazen Jukic, and M. Satyanarayanan. Openslide: A vendor-neutral software foundation for digital pathology. *Journal of Pathology Informatics*, 2013.

[11] John Goutsias and Henk J. A. M. Heijmans. Nonlinear multiresolution signal decomposition schemes—part i: Morphological pyramids. *IEEE TRANSACTIONS ON IMAGE PROCESSING*, 9:1862–1876, 11 2000.

[12] Roschzttardtz H., Grillet L., Isaure M.-P., Conejero G., Ortega R., Curie C., and Mari S. The plant cell nucleolus as a hot spot for iron. *The Journal of Biological Chemistry*, 32:27863–27866, 2011.

[13] L. Hou, D. Samaras, T. M. Kurc, Y. Gao, J. E. Davis, and J. H. Saltz. Patch-based convolutional neural network for whole slide tissue image classification. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2424–2433, 2016.

[14] Maximilian Ilse, Jakub Tomczak, and Max Welling. Attention-based deep multiple instance learning. *ICML*, 2018.

[15] Jewsbury, R., Bhalerao, A., Rajpoot, and N.M. A quadtree image representation for computational pathology. *In Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 648–656, 2021.

[16] Brendon Lutnick, Avinash Kammardi Shashiprakash, David Manthey, and Pinaki Sarder. User friendly, cloud based, whole slide image segmentation. In John E. Tomaszewski and Aaron D. Ward, editors, *Medical Imaging 2021: Digital Pathology*, volume 11603, pages 109 – 114. International Society for Optics and Photonics, SPIE, 2021.

[17] Muñoz-Aguirre M, Ntasis VF, Rojas S, and Guigó R. Pyhist: A histological image segmentation tool. *PLoS Comput Biol*, 10, 2020.

[18] Kingma D. P. and Welling M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[19] Liron Pantanowitz, Paul N. Valenstein, Andrew J. Evans, Keith J. Kaplan, John D. Pfeifer, David C. Wilbur, Laura C. Collins, and Terence J. Colgan. Review of the current state of whole slide imaging in pathology. *Journal of Pathology Informatics*, 2(1):36, 2011.

[20] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[21] Mark D. Zarella PhD, Douglas Bowman, Famke Aeffner DVM PhD, Navid Farahani MD, Albert Xthona, Syeda Fatima Absar MD, Anil Parwani MD PhD, Marilyn Bui MD PhD, and Douglas J. Hartman MD. A practical guide to whole slide imaging: A white paper from the digital pathology association. *Arch Pathol Lab Med*, 2:224–234, 2019.

[22] A. Ruifrok and D. Johnston. Quantification of histochemical staining by color deconvolution. *Analytical and quantitative cytology and histology*, 23:291–9, 2001.

[23] B. Sankur. Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic Imaging*, 2004.

[24] Kovalev Vassili, Diachenko Y, Malyshev Valery, Rjabceva Svetlana, Romaniuk Anatolii, Kolomiyets Olena, Lyndin Mykola, and Moskalenko Roman. Comparative features of open source software products for the development of an automated breast cancer diagnostic program. *Eastern Ukrainian Medical Journal*, 7, 12 2019.

[25] Punitha Viswanathan and Kalavathi Palanisamy. Analysis of file formats and lossless compression techniques for medical images. *International Journal of Scientific Research in Computing*, 2, 06 2020.

[26] Ross Wightman, Hugo Touvron, and Hervé Jégou. Resnet strikes back: An improved training procedure in timm. *CoRR*, 10 2021.

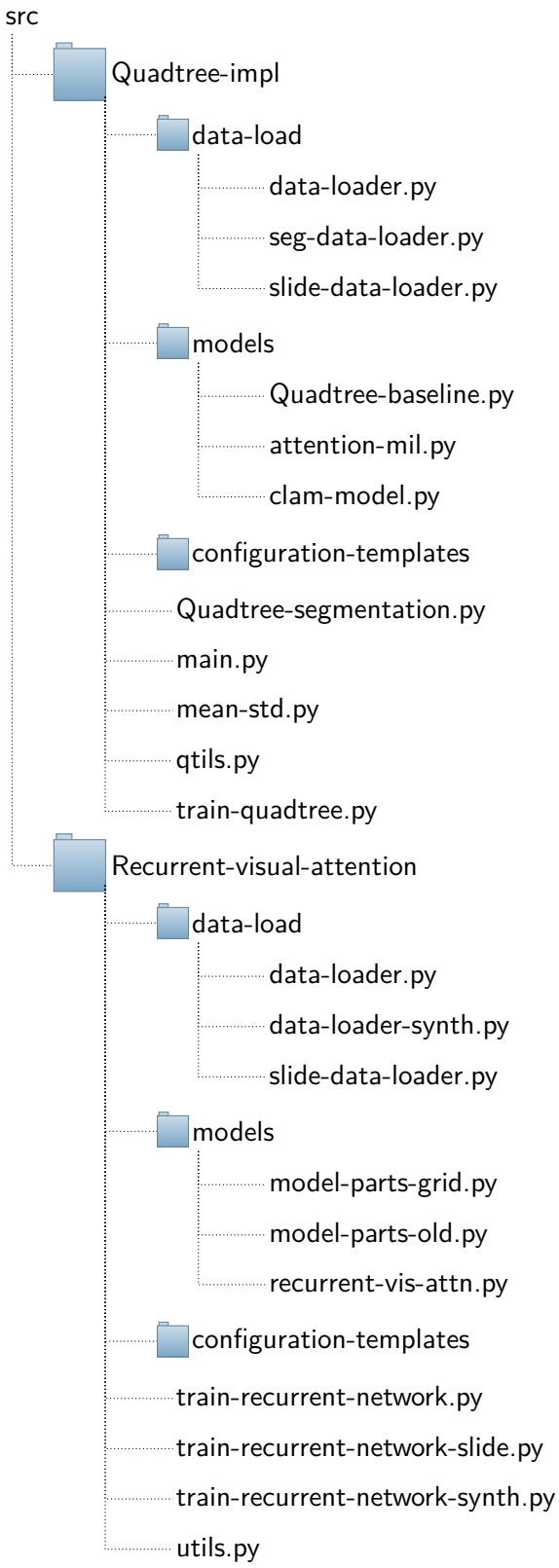
[27] Lu Ming Y. and et al. Data-efficient and weakly supervised computational pathology on whole-slide images. *Nature Biomedical Engineering*, 5:555–570, 2020.

Appendix A

Contents of attachment

The implementations of the Quadtree approach and the Recurrent visual attention model are available online through gitlab (https://gitlab.fel.cvut.cz/kralale4/methods-for-hierarchical-image-classification).

`src/Quadtree-impl/`	Implementation of the Quadtree method
`src/Recurrent-visual-attention/`	Implementation of the RVAM
`src/*/data-load/`	Different data loaders for each method
`src/*/models/`	implemented methods used in this work
`src/*/main.py`	runnable scripts for each method
`src/*/configuration-templates/`	templates for running training and testing experiments


src
Quadtree-impl
data-load
data-loader.py
seg-data-loader.py
slide-data-loader.py
models
Quadtree-baseline.py
attention-mil.py
clam-model.py
configuration-templates
Quadtree-segmentation.py
main.py
mean-std.py
qtils.py
train-quadtree.py
Recurrent-visual-attention
data-load
data-loader.py
data-loader-synth.py
slide-data-loader.py
models
model-parts-grid.py
model-parts-old.py
recurrent-vis-attn.py
configuration-templates
train-recurrent-network.py
train-recurrent-network-slide.py
train-recurrent-network-synth.py
utils.py