Universitetet
i Stavanger

Faculty of Science and Technology

Department of Electrical Engineering and Computer Science

MASTER THESIS

Study line / specialization: Master in engineering / Signal processing and robotics with medical technology	Spring semester 2023 Open
Author: Jon Tveit	
Main supervisor Hrafn Weishaupt Local supervisor Kjersti Engan Co-supervisor: Sabine Leh	
Title for thesis: Machine learning, unsupervised learning and stain normalization in digital nephropathology	
Study points: 30	
Subject words: Digital Pathology Stain normalization	Page numbers: 56 + appendix: 2 Stavanger 15. June 2023

Contents

Contents **i**

Summary **v**

1 Introduction **1**

1.1 Problem description 1

1.2 Aims and objectives 2

2 Background **4**

2.1 Chronic kidney disease 4

2.1.1 The kidney biopsy and digital histopathology 5

2.1.2 Glomeruli and morphological changes 5

2.1.3 Diagnosing CKD 7

2.1.4 Stain variation 7

2.2 Machine learning 8

2.2.1 Supervised learning 8
2.2.2 Unsupervised learning 9
2.2.3 Dimensionality reduction and cluster agreement . . . 11
2.3 Stain normalization 12
2.3.1 Reinhard stain normalization 12
2.3.2 Macenko stain normalization 12
2.3.3 Structure preserving stain normalization 13

3 Methods **14**

3.1 Overall approach 14
3.2 Data materials 15
3.3 Step 1: Stain normalization 16
3.3.1 Reference images 19
3.3.2 Reinhard stain normalization 20
3.3.3 Macenko stain normalization 21
3.3.4 Structure preserving colour normalization 22
3.3.5 Masking tissue of tissue background 23
3.3.6 Evaluating stain normalization results 25
3.4 Step 2: Image pre-processing and feature extraction 25
3.5 Step 3: Dimensionality reduction 26

3.6 Step 4: Clustering 26

3.6.1 Evaluating clustering results 27

4 Implementation **29**

4.1 Available hardware 29

4.2 Image reading 30

4.3 Implementation of stain normalization techniques 30

4.3.1 Reinhard implementation 30

4.3.2 Macenko implementation 31

4.3.3 SPCN implementation 31

4.3.4 Pipeline for normalizing images 31

4.3.5 Structural similarity index metric 32

4.4 Loading the pre-trained CNN and extracting features 32

4.5 Clustering 33

4.5.1 Dimensionality reduction implementation 33

4.5.2 Clustering implementation 33

4.5.3 Adjusted Rand Index 33

5 Results **34**

5.1 No normalization 36

5.2 Reinhard 36

5.3 Macenko 38

5.4 SPCN 40

6 Discussion **43**

7 Conclusions **45**

Bibliography **50**

Appendix **I**

Appendix A I

Appendix B IV

Summary

Chronic kidney disease is a serious health challenge and still, the field of study lacks awareness and funding. Improving the efficiency of diagnosing chronic disease is important. Machine learning can be used for various tasks in order to make CKD diagnosis more efficient. If the disease is discovered quickly it can be possible to reverse changes. In this project, we explore techniques that can improve clustering of glomeruli images.

The current thesis evaluates the effects of applying stain normalization to nephropathological data in order to improve unsupervised learning clustering. A unsupervised learning pipeline was implemented in order to evaluate the effects of using stain normalization techniques with different reference images. The stain normalization techniques that were implemented are: Reinhard stain normalization, Macenko stain normalization and Structure preserving color normalization. The evaluation of these methods was done by measuring clustering results from the unsupervised learning pipeline, using the Adjusted Rand Index metric. The results indicate that using these techniques will increase the cluster agreement between results and true labels for the data. Six reference images were used for each stain normalization technique. The average Adjusted Rand Index score for all reference images was increased using all three stain normalization techniques. The best performing method overall was the Reinhard stain normalization technique. This method gave both the highest single experiment and average score. The other normalization methods both have one score close to zero (unsuccessful clustering), and structure preserving color normalization would outperform the Reinhard method if this single clustering was more successful.

Chapter 1

Introduction

1.1 Problem description

In recent years, machine learning models have become accurate at detecting differences in images after being trained on large datasets [26].To train an image classification model you typically need datasets with labels that assign each image to a class [13]. With labels, the machine learning model can know whether the prediction it has made is correct or incorrect, and learn from it through a method called backpropagation [25]. This process is called supervised learning.

For some classification tasks, there are large labeled datasets available. In these cases, a deep learning classification model can be trained to achieve satisfactory classification accuracy [22]. In cases where there is less labeled data, trying to train a deep learning model can lead to overfitting and low classification accuracy [22]. In cases where the amount of annotated data is insufficient, we can attempt to use unsupervised learning techniques. In these techniques the model separates the data based on features, but does not go through the same training process.

The current thesis explore machine learning applied to medical images in the field of non-neoplastic nephropathology. Nephropathology is the study of kidneys diseases. In nephropathology there is a shortage of publicly avail-

able datasets with annotations. Annotations for nephropathology images are time consuming to obtain and require expert knowledge in the field of study [29]. In order to still use machine learning to separate data into classes, we implement a clustering pipeline. A similar approach was implemented for a different dataset by Sato et al. [26]. The dataset provided for this thesis contains 9088 images of glomeruli (kidney filtration units). Although this dataset is labeled, it is only used in this thesis to validate clustering results.

Clustering is a form of unsupervised learning and can be applied to unlabeled data [16]. By using various data processing techniques, images can be used as input to a pipeline that returns a cluster label. By implementing a process for automated clustering of data, it can be used to evaluate the impact that different pre-processing steps have on the final clustering. The specific step we want to evaluate in this project is stain normalization techniques. By implementing several stain normalization techniques in the clustering pipeline, we can evaluate how the stain normalization affects clustering results and whether and which stain normalization technique should be chosen.

Stain normalization is used to normalize the color scheme of images [24]. Ideally, the structure in the image is also preserved during this process. Staining refers to the chemical process of giving different tissue components different colors. Different staining chemicals are used to do this. Along with other factors such as tissue processing and section thickness this leads to color variations in glomerular images that might mislead machine learning algorithms. Kanwal et al. did a comprehensive review of the performance for different stain normalization techniques in [12].

1.2 Aims and objectives

The goal of this thesis is to see what effect different stain normalization techniques have on the final clustering result. Several stain normalization techniques will be implemented as well as all the steps involved in clustering of input images. Stain normalization is done using a reference image with color features that we want to transfer to another image. This thesis includes experiments using several different reference images for each stain

normalization technique.

To evaluate the results from the experiments that are explained in this thesis, a metric called the adjusted rand index is used [7]. The metric is used to evaluate the similarity between two clusterings [7]. By using it to compare the value assigned by the clustering model to the true class labels, it is possible to evaluate how good the clustering of glomerular images is.

The metric will tell us about how successful the algorithm was at differentiating between classes of glomerular lesions. The task of the algorithm is to cluster glomerular lesions into the following two classes: global sclerosis and everything else. The classes have different levels of morphological changes, and the data in the global sclerosis class are in the final stage of morphological change. These clustering targets are based on the features of the classes. The class global sclerosis shows distinctly different features from other classes and is more likely to separate from the rest in clustering.

In the context of a fully automated process for clustering of whole slide images, we want to evaluate the following questions: Is stain normalization a useful step for the final result? If so, what kind of stain normalization should be used? Finally, how does the choice of reference image affect the clustering results?

Chapter 2

Background

This thesis is part of work package 8 "Classification of glomerular lesions" of the strategic research project "Pathology services in the Western Norwegian Health Region – a center for applied digitization" (PiV).

2.1 Chronic kidney disease

Chronic kidney disease (CKD) is a group of heterogeneous diseases affecting the kidney [14]. In CKD the kidney loses function over time. CKD is defined as: "Abnormalities of kidney structure or function, present for >3 months, with implications for health" [23].

CKD include a variety of diseases. Examples are different types of inflammation, familial diseases, and chronic damage due to systemic diseases such as high blood pressure or diabetes [30]. CKD is both underrated and an unrecognized disease. The global prevalence of CKD is about 10% [3], which is in the same range as cardiovascular disease, diabetes mellitus, cancer and chronic respiratory diseases [28]. Vanholder et al. in [28] states that CKD is projected to be the fifth leading cause of death worldwide by 2040.

CKD is not a low-cost disease and underestimation and lack of awareness certainly contribute to the high cost of treatment. It is estimated that health

care costs for CKD are at least in the same range as for cancer and diabetes [28]. For patients, CKD means reduced quality of life and premature death.

The role of pathology is to suggest a specific diagnosis and provide information on chronicity and potential reversibility of changes. Pathology investigation of a kidney biopsy renders a correct diagnosis, predicts prognosis and guides treatment.

2.1.1 The kidney biopsy and digital histopathology

A kidney biopsy is a thin cylindrical piece of kidney tissue taken with a thin needle under local anesthesia. The biopsy is put in a fixation solution to preserve the tissue, and is then processed in a pathology laboratory. This process involves various means as shown in [12], including sectioning into thin slices, staining using chemicals and scanning with a digital slide scanner, resulting in a whole slide image (WSI). WSIs are large images with a lot of information. Through digital pathology, pathologists or machine learning algorithms can find important regions and features to study. The data that is used in this thesis are smaller regions extracted from WSIs.

2.1.2 Glomeruli and morphological changes

The kidney cortex consists of three compartments: Glomeruli, tubules and vessels. A glomerulus (glomeruli in plural) is a specialized bundle of tiny blood vessels or capillaries with very thin walls. Primary urine is formed when the blood is filtered through these thin capillary walls. This is why glomeruli are also referred to as filtration units [20]. The urine is collected in the Bowman's space which is surrounded by the Bowman's capsule. Figure 2.1. A shows a normal glomerulus with capillary convolute, Bowman's space and Bowman's capsule. Several glomeruli can be found in one nephropathology WSI. From the Bowman's space the urine enters the tubules. The tubules transport the urine and both absorb some of the filtered substances from the glomerulus and actively excrete other substances. Between the tubules, there is the interstitium, a narrow space with small blood vessels and small amounts of connective tissue.

Each of the three compartments can develop a variety of lesions which impact the kidney function. This thesis focuses on morphological changes in the glomerulus. Through digital nephropathology, morphological changes in the glomerulus can be observed and used to diagnose chronic kidney disease.

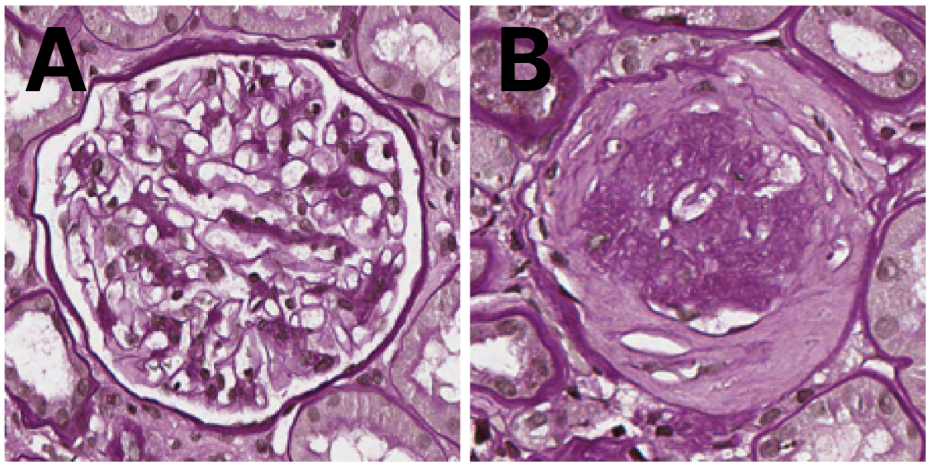

Figure 2.1: Glomerular patches, image A is a healthy glomerulus and image B is a scarred glomerulus (with morphological changes)

Glomeruli can develop a diversity of lesions that can affect the capillary convolute as well as the Bowman's space and capsule. Some examples of such lesions are necrosis, hypercellularity and sclerosis [6]. These lesions are examples of morphological changes in the glomeruli. The glomerular morphological changes are deviations from a normal glomerulus that can be measured. There are definitions that define what kind of morphological changes will indicate different lesions [6]. The morphological changes of the glomerulus that indicate global sclerosis is what we want to cluster using machine learning. Global sclerosis is the final stage of scarring in the glomerulus, where the entire glomerular structure is replaced by connective tissue and all capillaries have disappeared [11]. At this stage the glomerulus does not produce any more urine and is non-functional. Figure 2.1.B shows global sclerosis, the morphological change that stands most out of the classes. The glomerulus is completely scarred in this example. Counting the number of globally sclerosed glomeruli in a kidney biopsy will indicate the functional state of the kidney. Therefore we want to cluster between the class global sclerosis and everything else.

2.1.3 Diagnosing CKD

When a nephropathologist investigates a kidney biopsy, they will look at every glomerulus and try to classify the kind of lesion. The method of visual inspection is qualitative and is challenging to reproduce [13]. Visual inspection is also slow. if parts of it can be automated the pathologists will be able to diagnose patients faster. A deep learning classification model trained for this purpose would be useful for pathologist. The problem with using deep learning classification models is the lack of annotated data in the field. Another option for applying machine learning to nephropathology is to use unsupervised learning. In this thesis we want to help pathologists by improving automated tools that can cluster between global sclerosis and all other glomeruli.

2.1.4 Stain variation

Tissue extracted from biopsies are put through a process before they can be scanned and digitized, there are many steps in this process but the one we will be focusing on is staining. Staining means to dye the tissue using chemicals to highlight the present structures [12]. The use of different chemicals for staining tissue leads to varying color schemes for different images. Color variation may also be caused by other factors. The most commonly used stain is Hematoxylin and Eosin, however others are used as well such as PASM, which highlights morphological changes. The data used in this thesis is stained using Periodic Acid Schiff (PAS) [4]. There are many different factors that can alter the result when obtaining digital nephropathological images. In Kanwal et al. [12] they go through what kind of image artifacts that can be introduces in each step from biopsy to WSI. These artifacts cause stain variation in the final WSI [12], some of them are color related and can be processed with stain normalization.

One of the challenges in the process of automatic clustering of images from glomerular lesions is color variation. Variations in the stained sections of glomeruli might have a negative impact on clustering results. The stain variations introduces features in the images that might mislead machine learning models from focusing on the correct features. By using stain normalization, we can make the slices more similar in color and in that way

improve the accuracy of the machine learning model.

2.2 Machine learning

Machine learning is a broad term of algorithms for data analysis. IBM explains the term as follows: [8] "Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy." The next paragraphs will explain some branches of machine learning that are relevant for the current project.

2.2.1 Supervised learning

Supervised learning is the process of training machine learning models using data with labels [13]. Labels allow the machine learning model to compare the prediction of a data point to the true label and calculate the error. The error can be propagated back through the network to determine the contribution of the individual weights and adjust the weights to improve future predictions from the network [25]. This procedure is referred to as backpropagation, an iterative process that improves the model through training [25]. Most supervised learning networks are feed forward networks. A feed forward network is a one directional network [19]. To train a feed forward network, we use backpropagation [9].

Artificial neural networks (ANN) are the ground building blocks for deep learning. Figure 2.2 shows an example of a basic ANN. ANNs for supervised learning are typically made of layers and nodes [19]. A node is associated with a weight and a threshold. When the output of a node exceeds the thresholding value, it will activate and will send the data to the next layer [9]. To activate a node, the conditions for a non-linear function needs to be fulfilled. The inputs are outputs from nodes in the previous layer, with associated weights and biases that determine the significance of the respective input. A ANN uses input layers, hidden layers and output layers [19]. A fully connected layer is a layer, where the inputs for all neurons is connected from all neurons in the previous layer.

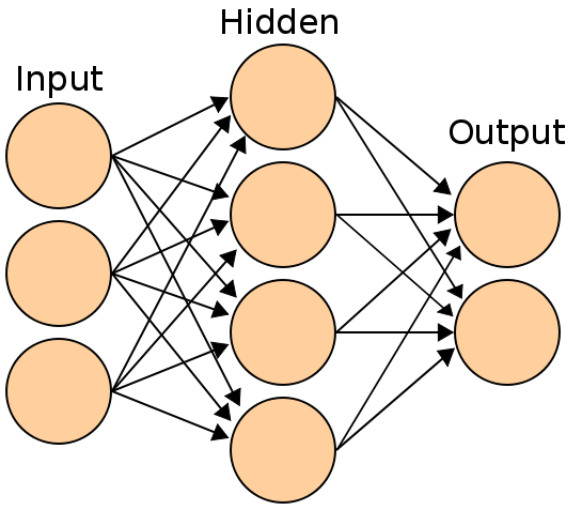

Figure 2.2: Artificial neural network, figure illustrating neurons in different layers. The figure is reprinted in unaltered form from Wikimedia commons, File:Artificial neural network.svg, licensed under CC BY-SA 3.0.

2.2.2 Unsupervised learning

Unsupervised learning is a branch of machine learning, where the algorithm is able to make a prediction on data without being trained using labeled data [13]. Some popular examples of unsupervised learning techniques are clustering and anomaly detection [8]. Algorithms like these are made to discover hidden patterns or similarities without human interaction.

Clustering

Clustering is the process of grouping similar data-points without labels together based on their features [16]. The objective of clustering is to divide the data into groups (clusters) where the points are similar to each other, and dissimilar to other clusters [2]. Clustering is used in unsupervised learning to automatically cluster data, results from unsupervised learning can also help discover features in the data that have previously been overlooked [13]. In this thesis we use Gaussian Mixture Models (GMM) for clustering, a probabilistic clustering model. A GMM is a clustering method where the

assumed distribution of data is a mixture of gaussian distributions. The model is fitted to the data in an iterative process, giving a model that can be used to predict the cluster for a given input [8].

Convolutional neural network

A convolutional neural network (CNN) is a deep learning model that is primarily used for image-based pattern recognition tasks [19]. This type of network is a form of ANN, but use additional types of layers. CNNs commonly have four main types of layers, first the input layer holds all the pixel values of the input image. Between input and output there can be a combination of convolutional and pooling layers [19], this is the part of the network responsible for feature extraction. The convolutional layers have an activation function that activates depending on if conditions are met while the pooling layers performs down-sampling, reducing the amount of parameters [22]. Lastly, the classification part of the model commonly consists of one or more fully connected layers, the output from these layers are the classification results [19]. As with other supervised deep learning algorithms the network can be trained on labeled data. By comparing the true labels to the predicted labels, the error can be calculated and used to estimate the individual weight's contribution to that error [22].

A pre-trained CNN refers to a network that has been trained on a different dataset, this can be useful if the dataset that is going to be classified is too small to saturate the training, or training the network is too computationally expensive. For this project, the CNN used is NasNetLarge. Without classification layers, the model has 1040 layers, consisting of mostly convolutional layers, pooling layers and activations. The model is pre-trained on a large dataset called ImageNet. ImageNet is a freely available dataset of more than 14 million hand-labeled images in over 21000 classes at the time of writing [10]. The advantage of using a pre-trained network is that the weights are already tuned to detect common features in data. Even though the dataset it was trained on is substantially different from the nephropathological data we use here, the model has learned about general image features. This process is called transfer learning.

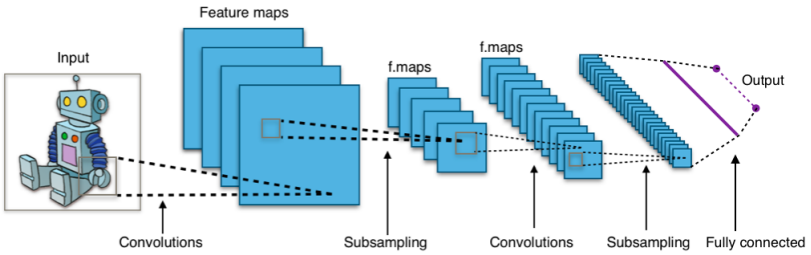

Figure 2.3: The figure illustrates a typical CNN with convolutional layers, pooling shown as subsampling and fully connected layer for classification. The figure is reprinted in unaltered form from Wikimedia commons, File:Typical cnn.svg, licensed under CC BY-SA 4.0.

CNN blocks

For the purpose of this project, a CNN can be divided into two blocks, convolutional blocks performing feature extraction (FE) and fully connected layers (FC) performing classification. The FC block refers to the fully connected layers at end of the model and the rest of the model belongs to the FE block [22].

Even though we have labels available, we only want to use them to validate unsupervised learning results. In this thesis, clustering will be used to separate images from different classes instead of using the FC block. This can be done by only putting the data through the FE part of the neural network and utilizing the feature vector output as the input to a clustering model. Before feeding the data to the clustering model, a dimensionality reduction technique will be applied.

2.2.3 Dimensionality reduction and cluster agreement

When using a CNN as a feature extractor, the dimensionality refers to the number of features extracted from the output of the CNN. The output data is going to contain more detailed features from input data, but the data is high dimensional and there is no meaningful way to visualized it without reducing the data points to a lower dimension. In this case we can use dimensionality reduction techniques to reduce the data to two-dimensional

data points. This reduction can be done by the Uniform Manifold Projection Algorithm (UMAP), an algorithm that projects the high dimensional data points into a lower dimension, where points that are close in the high dimension will also be close in the lower dimension [17]. After dimensionality reduction, the pattern that clustering will be performed on can be visualized in a two-dimensional plot and it might be possible to observe the difference between potential clusters.

2.3 Stain normalization

There are many different ways to perform stain normalization. A common factor is that the algorithm uses a reference image to obtain the desired color scheme and attempts to transfer this to other images. Following is an introduction to the three normalization techniques that were chosen for this project. The choice of techniques were based on what is commonly used as well as the results of Kanwal et al. in [12].

2.3.1 Reinhard stain normalization

Reinhard color normalization is one of the most widely used techniques for normalizing the color of pathological images. The normalization method is simple to implement, with few steps. Although the method is widely used, [24] shows that is does not perform as well as many of the other color normalization techniques that were evaluated in this study. Although the metrics that are presented in [24] apply to similarities between the original image and the processed image and might not impact clustering in the same way.

2.3.2 Macenko stain normalization

The Macenko color normalization method is a technique that does the color transformation of an image in the LAB color space. It was first shown in [15] and has become widely used in digital pathology and medical imaging. The method separates the color information from the brightness information.

These components are used to deconvolve the image into multiple stain contributions. The stain contributions are used to normalize the color vector so it becomes a consistent distribution across the image [26]. The target colors are based on a reference image, both the reference image and the source image are masked to remove white pixels. The normalized color vector from the reference image is used to reconstruct the original image, giving a normalized image with colors transferred from the reference image to the original image.

2.3.3 Structure preserving stain normalization

Structure preserving color normalization (SPCN) is a technique developed by Vahadane et al. in [27] for the purpose of reducing stain variation in histological images. The technique was further improved in [21], which removed some color artifacts. It also greatly improved the efficiency of the algorithm. The technique uses the stain density map from the source image and blends it with stain color basis from the reference image. Using this method, the structure of the image is preserved, while only the colors are changing [27].

Chapter 3

Methods

3.1 Overall approach

The following are the main steps that were used to obtain clustering results for glomerular images, illustrated in Figure 3.1. Step 1: The entire dataset was processed using different combinations of stain normalization techniques and reference images. Step 2: A pre-trained CNN was used to extract high-dimensional features for each image. Step 3: Dimensionality reduction was applied to reduce the features to two-dimensional data points. Step 4: A clustering algorithm was used to determine the decision boundary between clusters.

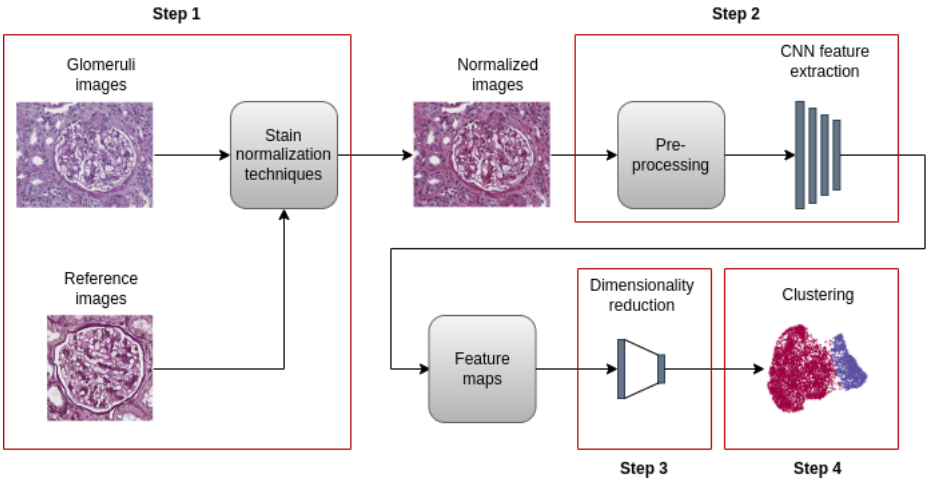

Figure 3.1: Machine learning approach from image dataset to classification results

3.2 Data materials

The dataset includes images from the work of Weis et al. [4] and contains 9088 anonymized images of single glomeruli. All the images in this dataset are obtained from Periodic Acid-Schiff (PAS) stained kidney biopsy sections. In this data there are ten different classes of glomeruli with nephrological disease, as well as normal glomeruli. The classes are distributed with the following class labels: 4347 normal glomerulus, 394 amyloidosis, 233 DM nodular, 1954 global sclerosis, 946 mesangial expansion, 69 MPGN, 454 necrosis, 174 NOS, 504 segmental sclerosis and finally 13 TX glomerulitis. At the time of writing, this dataset is not publicly available. The data was made available for this thesis through the research group (PiV). Images are in TIFF format and have variable image size, mostly around 220 pixels each direction +-50 pixels.

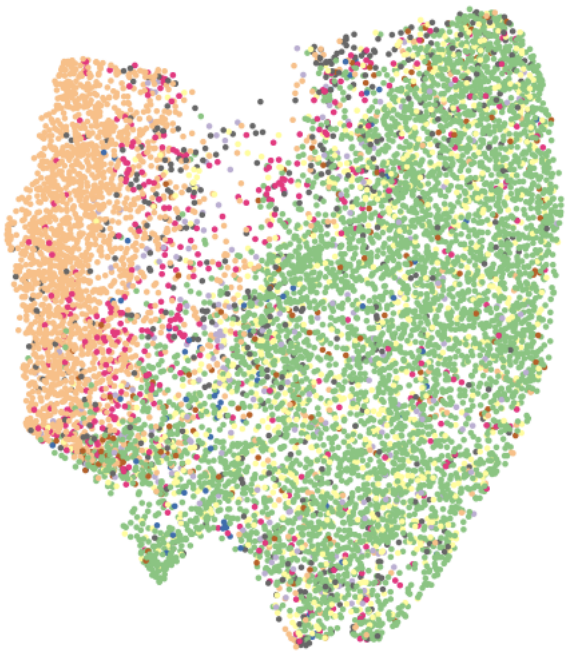

Figure 3.2: Class distribution for ground truth labels after processing, the orange class here is global sclerosis and green is normal

Class labels were assigned by a pathologist in [4], and are considered the true class labels for this thesis. After extracting features from the data and performing dimensionality reduction (step 2 and 3 in figure 3.1, we can plot the data distribution with the true class labels with different colors representing the different classes. From figure 3.2 we can see that the distribution of data in one of the classes (this is global sclerosis) separates from the rest, while other classes are more spread.

3.3 Step 1: Stain normalization

In this project, different stain normalization techniques were applied to the images to compensate for colour variations and evaluate if this process will improve automatic clustering of glomerular lesions. Figure 3.3 shows an example of stain normalization for one image. The image that is going to

be normalized it referred to as the source image, and the reference image is referred to as target image.

In [24], several stain normalization techniques were compared and from the results of their evaluation, the best performing methods were selected for implementation: Structure Preserving Color Normalization (SPCN) and Macenko stain normalization. The Reinhard method was also chosen, due to being one of the most widely used stain normalization techniques. The method with the highest overall score from the review [24] was SPCN. This method was described as preserving brightness and structure well, while introducing minimal artifacts.

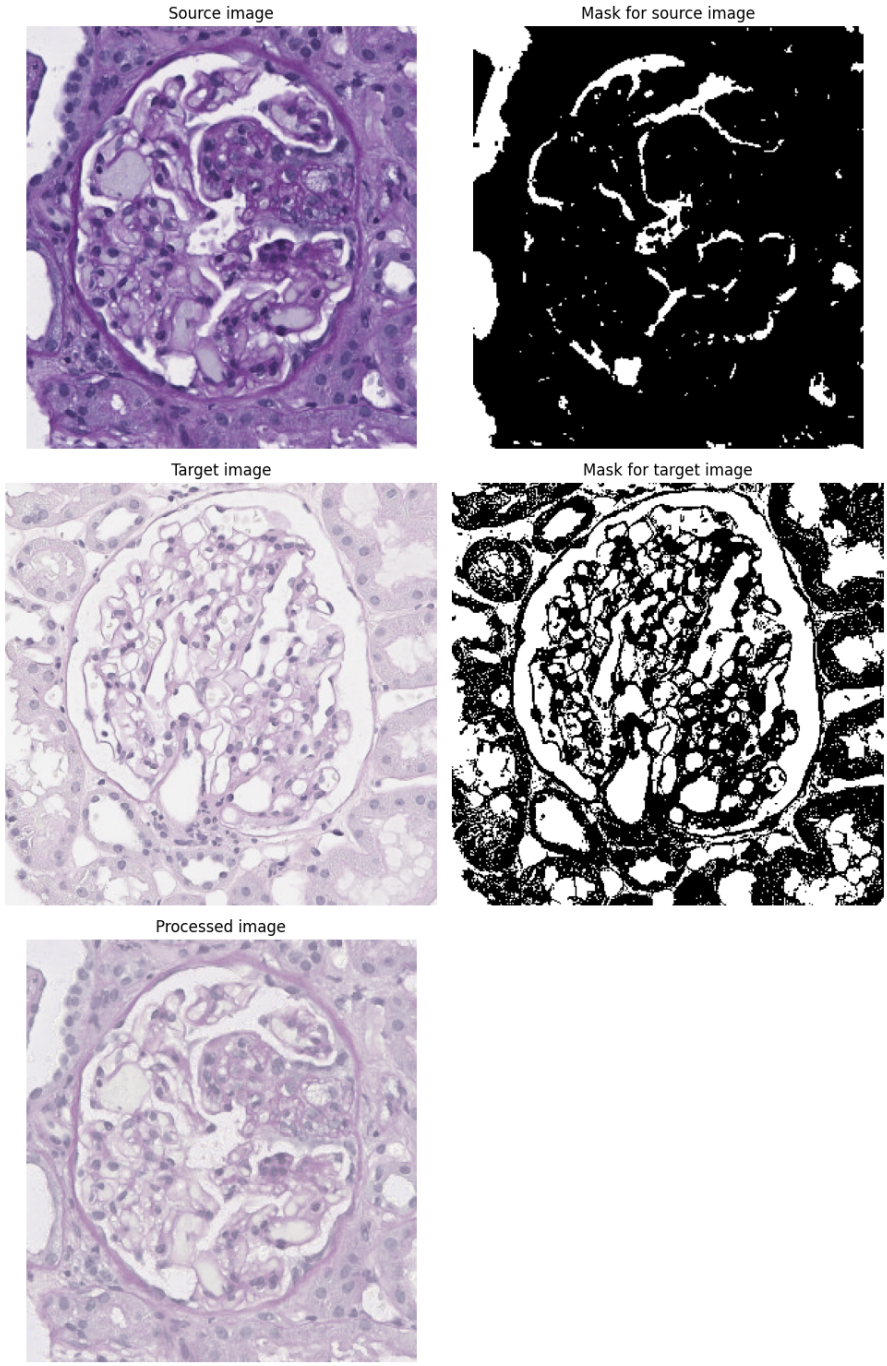

Figure 3.3: The figure illustrates stain normalization for one image with white pixel masks for both source image and target image, using the Reinhard method.

3.3.1 Reference images

When normalizing the images, the reference image is used to specify the desired color scheme of the output image. No golden standard for choosing a reference image appears to exist in the literature. Some things that were taken into consideration were stain intensity, the amount of background present in the image and the morphological changes. The task of selecting reference images was done by pathologist Sabine Leh. The pathologist chose three reference images based on stain intensity/brightness (low, perfect, high) and three images based on the morphological changes, where the selected glomerular lesions were global sclerosis, amyloidosis and necrosis. The reference images are from the dataset of labeled glomeruli [4]. Figure 3.4 shows the six reference images.

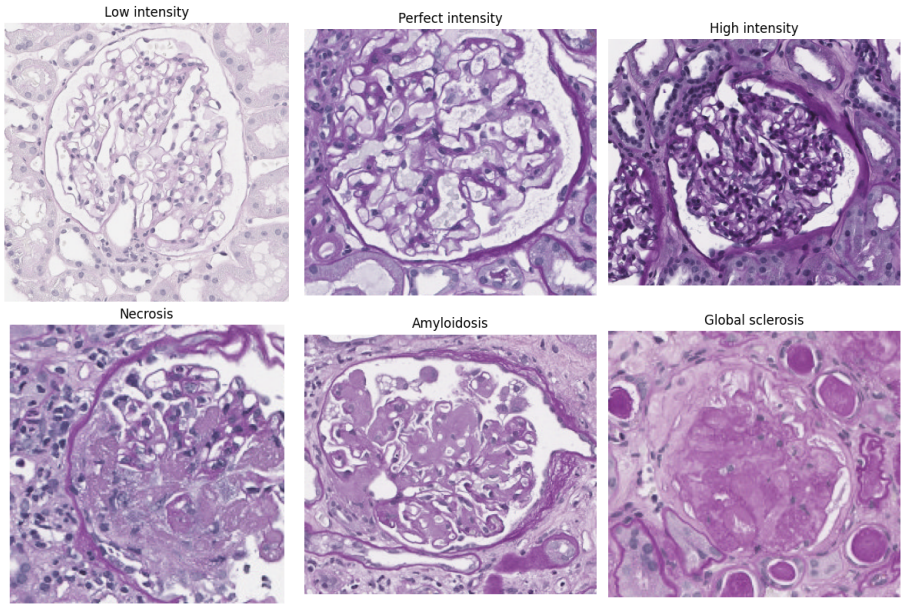

Figure 3.4: The reference images that were used for all stain normalization techniques

3.3.2 Reinhard stain normalization

Because Macenko and SPCN are complicated methods that are difficult to illustrate, a full explanation of the Reinhard method is used to illustrate stain normalization. Before processing images using the Reinhard method, white pixel masks are applied to both the target and source images. The steps for performing Reinhard stain normalization are as follows: First load the source and target images and convert both images to LAB color space. Second, process each color channel using the formula in equation 3.1.

$$\hat{I}_{prc} = \frac{I_{src} - \mu_{I_{src}}}{\sigma_{I_{src}}} \cdot \sigma_{I_{\text{trg}}} + \mu_{I_{\text{trg}}} \tag{3.1}$$

The equation uses the following abbreviations: prc for processed, src for source and trg for target. The source image is the image we want to normalize, the target image is the reference image, and the processed image is the resulting normalized image. μI represents the color mean and σI represents the standard deviation of color in the respective image. μI and σI are calculated from the target image colors after the mask to remove white pixels has been applied. μI and σI are calculated for all pixel values.

The calculation is iterated three times, once for each color channel. After the iterations are complete, the image is converted back to RGB color space and can be evaluated and used for further image processing. The Reinhard method is expected to introduce artifacts in the processed image in the form of blurring [24]. An example of an image processed with the Reinhard method, using all six reference images is shown in figure 3.5.

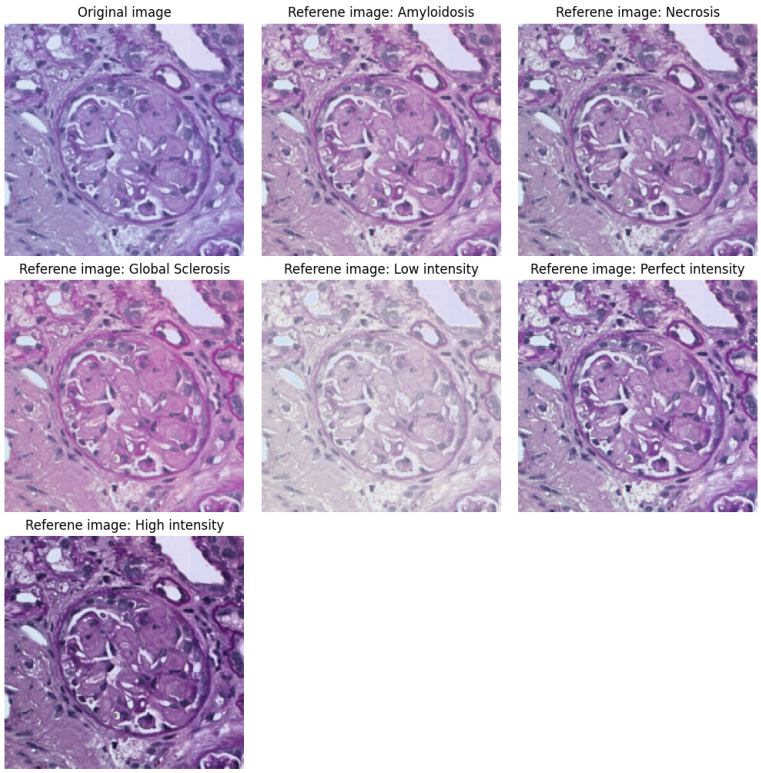

Figure 3.5: The figure shows an original image, and the same image normalized with the Reinhard method using the six reference images show in figure 3.4

3.3.3 Macenko stain normalization

The Macenko stain normalization uses stain vectors to extract the dominant colors in the reference image. The colors are then separated from the structure to obtain stain intensities for the pixels. The stain intensities are then used to normalize the colors of the target image. [15] In [24], the results from using Macenko stain normalization are described as inconsistent and deviating too much from the target image. Like in the Reinhard method, white pixel masks are used for both target and source images. The detailed steps in the Macenko methods are complicated and will not be covered here. The method is well presented in [24]. An example of an image processed with the Macenko method, using all six reference images is shown in figure 3.6.

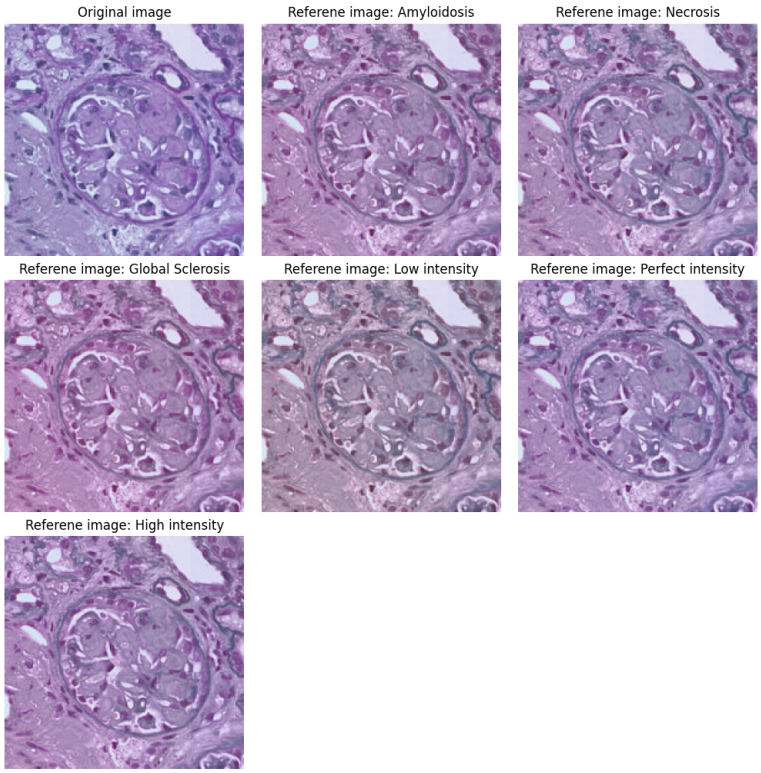

Figure 3.6: The figure shows an original image, and the same image normalized with the Macenko method using the six reference images show in figure 3.4

3.3.4 Structure preserving colour normalization

SPCN aims to normalize stain images without distorting the underlying structure. The method operates via the following steps: First read the target and source image. Second, the image is represented as a stain density map and a stain color appearance matrix [24]. Third, an unsupervised stain separation method, i.e. non-negative matrix factorization is applied. Step four is to transfer color from the source image to the target image by combining the color appearance matrix of the source image with the stain density map of the target image [24]. White pixels do not need to be removed in SPCN, and no masks are used. An example of an image processed with the SPCN method, using all six reference images is shown in figure 3.7.

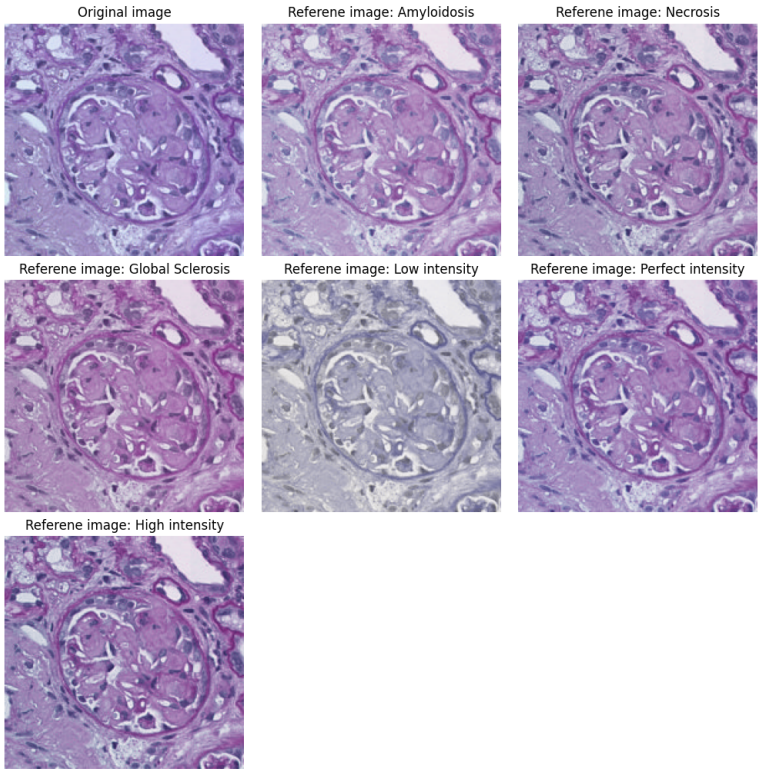

Figure 3.7: The figure shows an original image, and the same image normalized with the SPCN method using the six reference images show in figure 3.4

3.3.5 Masking tissue of tissue background

For both the Reinhard and Macenko color normalisation techniques, white pixels should be removed during processing to get accurate results. An example of how a mask used to remove white pixels looks is illustrated in figure 3.8. Obtaining a white pixel mask was done using thresholding in the Hue, Saturation and Intensity (HSI) colour space. After converting the respective image to HSI space from RGB, each HSI component was thresholded individually using parameters hue-min: 0, hue-max: 1, saturation-min: 0, saturation-max: 0.2, intensity-min: 220, intensity-max: 255. The HSI masks are then combined by setting the new mask value to one in areas where at least one of the masks has a value of one. These parameters can be found under saliency/cellularity-detection-thresholding in the documenta-

tion for [1] on github. For both the Macenko and Reinhard methods, white pixel masks were used for both the reference image and the input image. With these masks applied, the normalization techniques will only affect the foreground (tissue) parts of the image, and the amount of background will not affect the colors of the processed image.

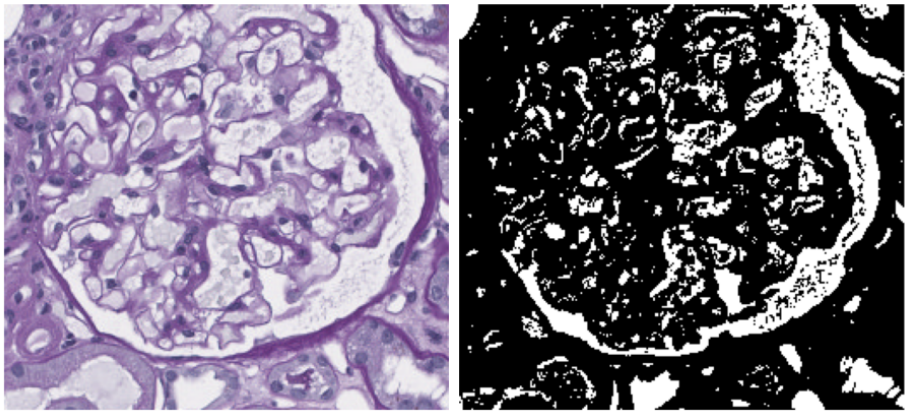

Figure 3.8: Example of a white pixel mask from a reference image

Figure 3.9 highlights the importance of using a mask to remove white pixels. The background for the target image has been normalized when not applying masks, leading to a image with a pink background that is supposed to be white.

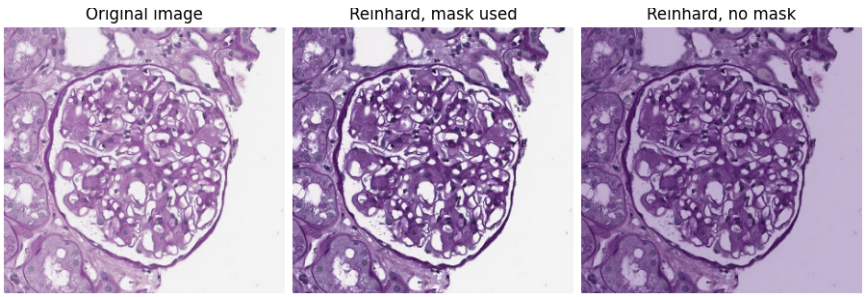

Figure 3.9: Illustration of Reinhard normalization using the amyloidosis reference image from figure 3.4, both with and without applying masks in the normalization

3.3.6 Evaluating stain normalization results

In this study we use different colour normalization techniques to minimize color variation between images without affecting other properties of the image. In order to know if this is the case, we will have to use some evaluation metrics to calculate the similarity of the original and the processed image. The images should also be manually inspected for artifacts that may form a pattern for the specific normalization technique. In [24], a comprehensive study for color normalization methods for histopathology images, the following metrics were recommended for evaluating color normalization: Structural Similarity index metric (SSIM), Quaternion Structural Similarity index metric (QSSIM) and Pearson correlation coefficient (PCC). As the main objective in this thesis is to evaluate the effect that the normalization techniques lead to in clustering, only SSIM was implemented. This metric is enough to give an idea of the structural similarity between a reference image and a processed image.

3.4 Step 2: Image pre-processing and feature extraction

After applying a stain normalization technique, the data is pre-processed to get the correct dimensions to be fed to the convolutional neural network (CNN). The CNN, NasNetLarge can use any symmetrical size bigger than 32 for inputs. A common input shape for CNNs was selected (224,224,3). The size of the images vary and the images are therefore resized to the appropriate size. The images are also converted to tensors, the input format expected by the CNN architecture.

A pre-trained version of NasNetLarge is used for the feature extraction. The reason that a pre-trained network was selected is because it takes a very long time to train a large network like this, and requires a lot of data to be able to get meaningful results. If a large network like this is trained on a small data set it would likely overfit the training data, meaning it could memorize the data rather than learning the necessary parameters to detect similar data. If the model is overfitted, it would lose the ability to generalize and classify yet unseen images. Since the model is not going to be trained,

all layers are set to non-trainable.

The CNN is used to extract features from all samples. Since the network is only used as a feature extractor, the classification layer of the network (FC) is removed, leaving a single feature vector per image as output from the network. Using images as input, the network will find properties of the image through a combination of different layers.

The CNN outputs a feature vector containing 4032 features for each image. Before it is used in a clustering model, the dimensionality reduction technique is applied.

3.5 Step 3: Dimensionality reduction

The output features from the convolutional neural network are high-dimensional and can be reduced to lower dimensions by using a dimensionality reduction technique. Based on [17], a technique called Uniform manifold projection algorithm (UMAP) was used. This method projects the higher dimensional data to a lower dimension. UMAP is a popular method for pre-processing high dimensional data for both visualization and clustering [26]. The algorithm uses the distance relation of high-dimensional data points to compute a lower-dimensional representation of the data. Data points that are closely related in the high-dimensional space will be projected with a corresponding relation in the low-dimensional space. Another dimensionality reduction technique, t-SNE was also implemented for evaluation, but was discarded due to worse performance.

3.6 Step 4: Clustering

There are several ways to do clustering. In this thesis, Gaussian Mixture Models (GMMs) are used. The project utilized GMM parameters identified through systematic testing on a similar dataset in another project previously conducted in the research group. Along with GMMs, k-means clustering was also tested in the early stages of this project but discarded as the results were far worse.

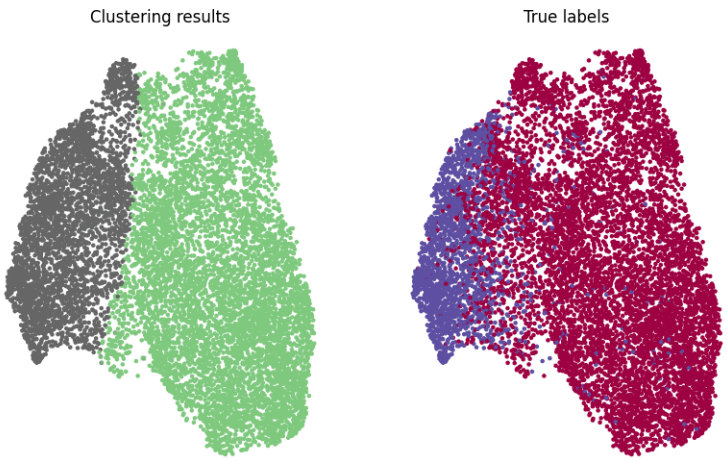

Figure 3.10: Clustering, the left figure shows results from clustering and the right figure shows true labels between two classes. The black/blue clusters are in this case global sclerosis and the green/red clusters are everything else

3.6.1 Evaluating clustering results

Using a method called the Adjusted Rand Index (ARI), clustering agreement can be calculated and used to validate clustering results [7]. The ARI is used to measure the agreement between two clusters, and is adjusted to account for cluster agreement that is introduced by chance. The ARI score is measured from -1 to 1, where a value of one is a perfect agreement [7]. Values closer to one indicate high cluster agreement between two compared clusters, and values closer to -1 indicate that the clusterings are completely different [18].

$$RI = \frac{TP + TN}{TP + FP + TN + FN} \tag{3.2}$$

Equation 3.2 describes the Rand Index (RI) when a cluster is compared to the true labels. Abbreviations in the equation indicate the number of elements that belongs to the following categories: TP for true positive, FP

for false positives, TN for true negatives, and FN for false negatives.

Equation 3.3 shows how to calculate the rand index between different clusters, where i and j represents the different clusters and n is the number of samples in each cluster [7].

$$\mathrm{RI} = \sum_{ij} \binom{n_{ij}}{2} \tag{3.3}$$

Equation 3.4 shows the adjusted RI, which is adjusted for chance. The max RI is the highest possible cluster agreement, 1 [7].

$$\mathrm{ARI} = \frac{\mathrm{RI} - \mathrm{Expected\_RI}}{\mathrm{Max\_RI} - \mathrm{Expected\_RI}} \tag{3.4}$$

The expected RI is calculated in equation 3.5, where i and j represents the different clusters, N is the total number of samples in the dataset, and n is the number of samples in each cluster [7].

$$\mathrm{Expected\_RI} = \frac{\sum \binom{n_i}{2} \cdot \sum \binom{n_j}{2}}{\binom{N}{2}} \tag{3.5}$$

Chapter 4

Implementation

All data processing in this project was done in Python3 considering the wide range of data processing libraries that are available. The work was done in JupyterLab notebooks which can give a good overview of variables and useful illustrations between different steps of the process. Various plots were used to evaluate that the steps were successful. The code for the plots can be seen in the appendix section but will not be covered in the implementation section. A lot of the code is dedicated to simplifying the process of running the same experiment for the different normalization methods and reference images. These section are also available in the appendix.

4.1 Available hardware

Some machine learning algorithms have the capability to utilize graphics processing units (GPU) for calculations. TensorFlow does support GPU processing, and this option was used for feature extraction. The GPU used was a Nvidia RTX 2070 with 8 gigabytes of video memory. By using this GPU for feature extraction, the processing time was effectively halved, as well as freeing up the CPU for other tasks. The other resource intensive task was stain normalization, which was done using the central processing unit (CPU). The CPU used was an AMD Ryzen 5 2600X with 6 cores. At 100% CPU usage, each normalization of the whole dataset lasted around

one hour.

4.2 Image reading

Images from the dataset [4] are in the format .TIFF. The images were read using the scikit-image function "io.imread" with three colour channels. These images have variable sizes, and will therefore have to be resized later on. This function reads the images as RGB image arrays, which can be used by other packages for further processing.

4.3 Implementation of stain normalization techniques

For the three stain normalization techniques, the steps for implementation will be different. The Reinhard and Macenko methods are implemented in a similar fashion as they are both included in the same python library, HistomicsTK [1] and both require white pixel masks to get accurate colour processing. To create the masks, the images were converted to HSI color space using a bespoke function that utilizes open-cv "inRange" which is a thresholding function with thresholding values found in the documentation of HistomicsTK. The following sections explain the specific color normalization for each normalization method.

4.3.1 Reinhard implementation

For the Reinhard method, the function "histomicstk.preprocessing.color-normalization.reinhard" was used. The function expects the following inputs: input image, color mean value and standard deviation for reference image, and mask for the input image. The function returns the normalized image. The mean and standard deviation for the reference image is calculated using another function from HistomicsTK, "htk.preprocessing.color-conversion.lab-mean-std". This function uses the reference image and the respective white pixel mask as input and outputs both mean and standard deviation.

4.3.2 Macenko implementation

The main function used for Macenko stain normalization is "histomicstk.preprocessing.color-normalization.deconvolution-based -normalization". This function requires the input image, a stain color matrix for the reference image, stain unmixing routines and the white pixel mask for the input image. The function returns the normalized image. The stain unmixing routines are provided in the documentation of histomicsTK, and the stain color matrix is calculeted with "histomicstk.preprocessing.color-deconvolution.stain-unmixing-routine" function, which uses the inputs: stain unmixing routines, reference image and the white pixel mask for the reference image and returns the stain color matrix.

4.3.3 SPCN implementation

SPCN is implemented quite differently than Reinhard and Macenko. Instead of using histomicsTK library, itk-spcn [5] is used for this method. While this implementation was designed for use with H&E images, it was still used in the current project, due to being the only publicly available implementation of SPCN that was found in the literature. Since the images in the dataset used in this thesis is stained using PAS, the method might not work as intended. The method was implemented anyway, and the SSIM score will tell us about the performance. The method does not require masks that remove white pixels. The itk toolkit has its own image reading function, "itk.imread" that imports the image with RGB color channels. The function "itk.structure-preserving-color-normalization-filter" is used to normalize the images. It uses input image and reference image and outputs the normalized image.

4.3.4 Pipeline for normalizing images

To normalize the whole dataset, two for loops are used to iterate through the data. The first loop goes through all classes, and for each class lists the directory and stores it in a variable used to obtain all filenames and the length of the class. The second loop iterates through all images in the

class. For each image in the class, the image is read, normalized with the respective method and saved in a numpy array. After all the images are normalized, which is a time consuming process, another loop is used to save the images within folders that describe the class.

4.3.5 Structural similarity index metric

SSIM was implemented to compare all normalized images to the original. To do this, a for loop going through each image was used. Calculating the SSIM was done by using the "skimage.metrics.structural-similarity" function. The function only uses the two images as input along with the number of color channels and outputs the SSIM score. The SSIM score that is presented in results is the mean SSIM for the whole dataset, calculated by using "numpy.mean" on the array containing all the individual SSIM scores.

4.4 Loading the pre-trained CNN and extracting features

Before loading the CNN, the input shape for images to the network was specified to (224,224,3). the network can then be loaded by using the TensorFlow function
"keras.applications.NASNetLarge(weights='imagenet',include-top=False, input-shape=input-shape). The model is loaded with the wights from pre-training on imagenet and the option to not include the top means to leave out the classification layer(s). The model is then set to non-trainable and a pooling layer is added to the output by using the function "keras.layers.GlobalAveragePooling2D". After these steps are finished, the model is compiled and is now ready to be used as a feature extractor.

The features are extracted using a for loop that iterated through all images in the dataset. For each image, the image is loaded, converted to an array and dimensions are expanded. The image is now a tensor and is used as input to the CNN, the CNN then returns the features for the image and is stored in an array. The shape of the feature array is (9088, 4032).

4.5 Clustering

4.5.1 Dimensionality reduction implementation

Dimensionality reduction was implemented using the "umap-learn" python library. To initialize the dimensionality reducer, the function "UMAP" is used with inputs: number of neighbors, minimum distance, spread, initialization, local connectivity, metric and number of components. The values for the parameters can be seen in appendix B. The function returns a dimensionality reduction model. To use the model, the function "reducer-umap.fit-transform(features)" is used. This returns dimensionality reduced features with the shape (9088, 2), which can be visualized in a two dimensional plot.

4.5.2 Clustering implementation

The Gaussian mixture model is initiated by using "sklearn.mixture.GaussianMixture" with the following input parameters: number of clusters, number of initializations and init parameters, these values can also be seen in appendix B. The function returns a clustering model, that is used to predict on the dimensionality reduced data by doing "clustering-model.predict(DimRed-features). Returning a binary array with predicted classes for each data point.

4.5.3 Adjusted Rand Index

To calculated the ARI, the function "sklearn.metrics.cluster.adjusted-rand-score" is used. The function uses true labels and clustering results as input. The true labels are found by simply iterating through all the filenames and adding a one if the filename contains "global-sclerosis" and zero otherwise. The function returns the ARI score.

Chapter 5

Results

The clustering results of the three different stain normalization techniques were compared to the results of the clustering without using stain normalization. The ARI is found by comparing true labels to the clustering results. The results are presented by clustering plots and the calculated ARI scores. The SSIM was also calculated and is included in the tables, although the score does not seem to have a close relation to clustering results. Figure 5.1 shows all ARI scores together, with all results for each reference image grouped together. The results show variation in performance based on both reference image and the applied stain normalization technique.

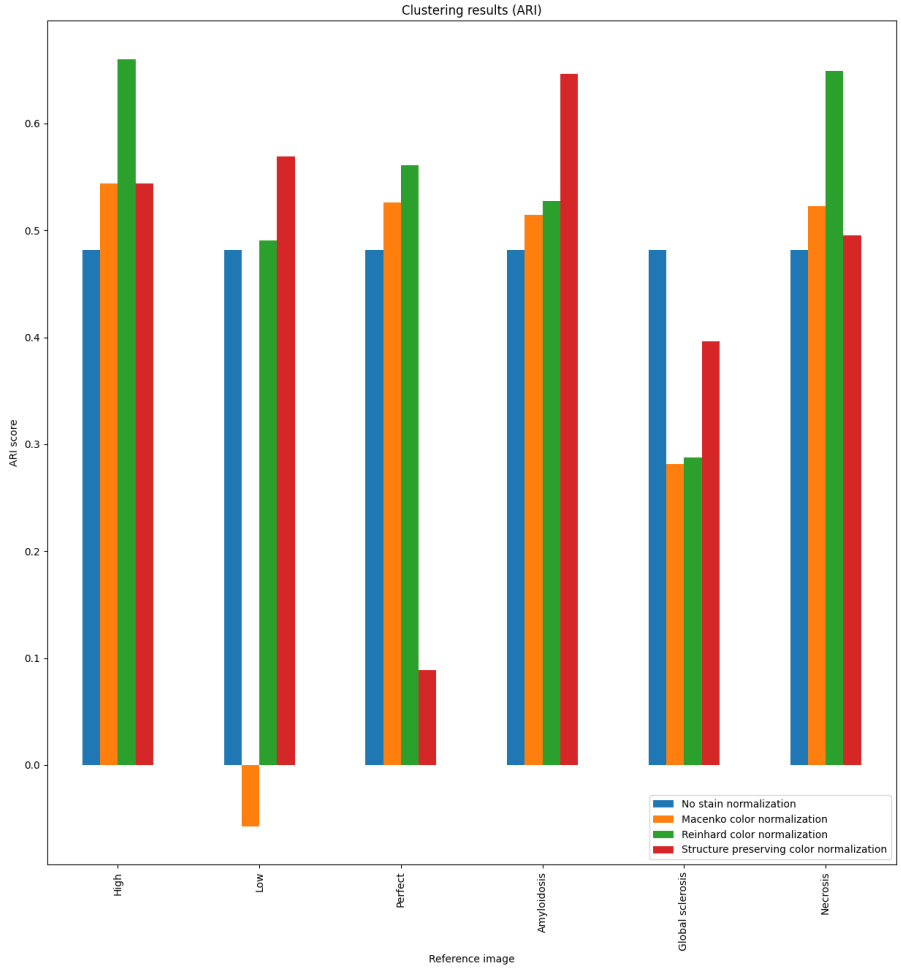

Figure 5.1: Bar plot of the ARI scores for the normalization methods and the different reference images.

5.1 No normalization

With no stain normalization applied to the data, the ARI was calculated to 0.48143. Figure 5.2 shows the clustering results along with the true labels.

Figure 5.2: Clustering results for data with no stain normalization. The left figure shows the cluster labels, where black is assumed to be global sclerosis. The right figure shows the true labels, where blue is global sclerosis and red all other classes.

5.2 Reinhard

In this subsection, the results obtained from using the Reinhard stain normalization technique are presented. Figure 5.3 shows the clustering results along with the true labels.

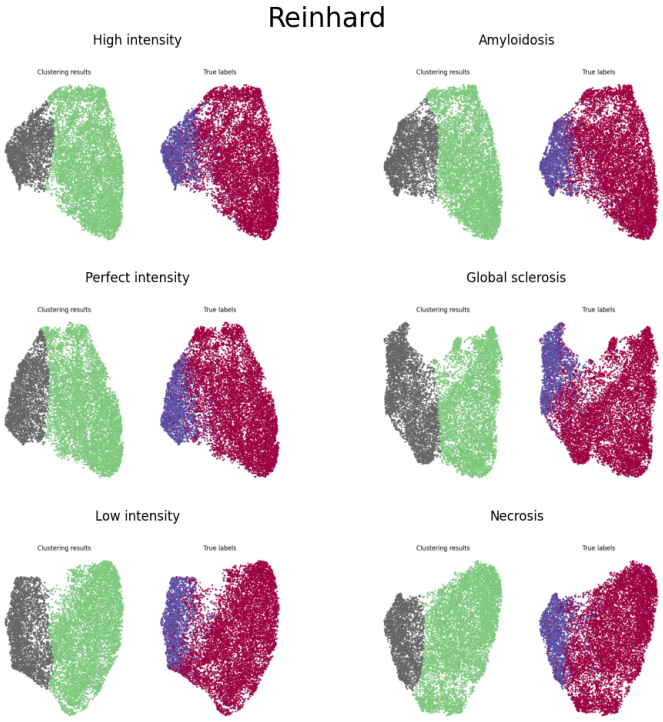

Figure 5.3: Clustering results for data processed with Reinhard color normalization. The left figure for each reference image shows the cluster labels, where black is assumed to be global sclerosis. The right figure for each reference images shows the true labels, where blue is global sclerosis and red all other classes.

Table 5.1: Comparison of SSIM and ARI values for Reinhard method

Category	SSIM	ARI
High	0.92278	0.66048
Low	0.98796	0.49095
Perfect	0.97606	0.56077
Amyloidosis	0.98242	0.52777
Global sclerosis	0.96930	0.28751
Necrosis	0.98310	0.64905
Mean values	0.97270	0.52942

Table 5.1 shows the ARI scores from clustering with Reinhard stain normalization. The results are relatively consistent, with no scores close to 0. The

highest ARI score in the Reinhard method was obtained using the high intensity reference image with a score of 0.66 and is actually the highest score of all methods. The necrosis reference image comes in at a close second with a score of 0.65. For the global sclerosis reference image, the ARI score is reduced to 0.28 after applying stain normalization. This image might be unfit as a reference image for stain normalization. Overall, the Reinhard method contains both the highest single and mean ARI score, with a mean score of 0.52942.

5.3 Macenko

In this subsection, the results obtained from using the Macenko stain normalization technique are presented. Figure 5.4 shows the clustering results along with the true labels.

Figure 5.4: Clustering results for data processed with Macenko color normalization. The left figure for each reference image shows the cluster labels, where black is assumed to be global sclerosis. The right figure for each reference images shows the true labels, where blue is global sclerosis and red all other classes.

The results of the Macenko stain normalization technique are shown in table 5.2 and has the lowest mean ARI score out of the three techniques, with a mean score of 0.39. This is largely due to a negative result in normalization using the low intensity reference image. If we exclude this result, the mean score of the remaining four will be 0.48. Three of the results are improvements over the data without normalization, but they are not reaching the high values seen in the Reinhard method. The result for the global sclerosis

Table 5.2: Comparison of SSIM and ARI values for Macenko method

Category	SSIM	ARI
High	0.99873	0.54373
Low	0.87605	-0.05712
Perfect	0.99857	0.52599
Amyloidosis	0.99873	0.51474
Global sclerosis	0.99718	0.28141
Necrosis	0.99839	0.52253
Mean values	0.97794	0.38854

reference image is very similar to the result in the Reinhard method.

5.4 SPCN

In this subsection, the results obtained from using structure preserving color normalization are presented. Figure 5.5 shows the clustering results along with the true labels.

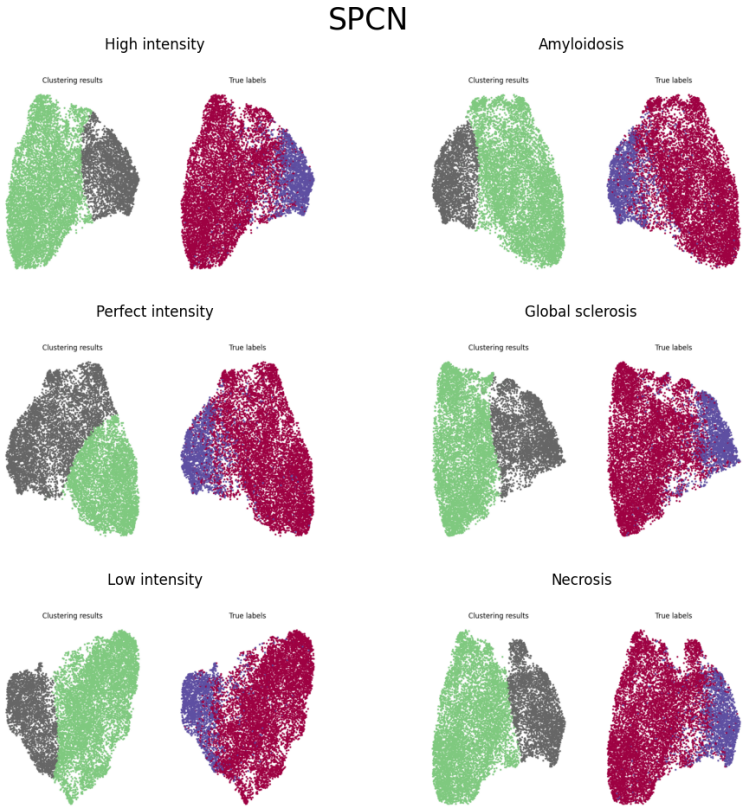

Figure 5.5: Clustering results for data processed with structure preserving color normalization. The left figure for each reference image shows the cluster labels, where black is assumed to be global sclerosis. The right figure for each reference images shows the true labels, where blue is global sclerosis and red all other classes.

Table 5.3: Comparison of SSIM and ARI values for SPCN method

Category	**SSIM**	**ARI**
High	0.97259	0.54416
Low	0.94221	0.56930
Perfect	0.96887	0.08878
Amyloidosis	0.95747	0.64681
Global sclerosis	0.97841	0.39621
Necrosis	0.97627	0.49553
Mean values	0.96597	0.45679

SPCN ARI results are shown in table 5.3 and have a mean value of 0.46, largely due to a very low score of 0.09 with the perfect intensity reference image. It is unexpected for this reference image to give such a low ARI score. Looking at the clustering in figure 5.5 - Perfect intensity, the decision boundary does not seem to be reflecting the features of global sclerosis. In the figure, separation between the clusters can be seen and it would likely only require minor adjustments to the clustering parameters in order to get a better result. If we exclude this result, the new mean ARI for the method is 0.53. If we assume the clustering using the perfect intensity reference image was 0.53 or higher, then SPCN would actually be the best overall performing method out of the three (with corresponding exclusion of negative result in the Macenko method). Notable results from SPCN is the ARI score from using the amyloidosis reference image, with a relatively high score of 0.65.

Considering that the method is optimized for H&E stained images instead of PAS stained images, the method holds up well. The experiments from SPCN show the lowest mean SSIM scores, although by a tiny margin. It was expected from the results of [24] that SPCN would give the highest SSIM scores, indicating that the method is not entirely compatible with PAS stained images.

Chapter 6

Discussion

For the Macenko method with the low intensity reference image, and the SPCN method with the perfect intensity reference image, the results were poor. The same images lead to better results in other methods, as shown in figure 5.1. This might not indicate bad reference images, rather than a "hit-or-miss" for the clustering. The parameters for the clustering model was after all not optimized for each stain normalization technique, and remained unaltered for all experiments in order to not introduce bias. It is possible that optimizing the clustering algorithm for each stain normalization would improve performance, and more strongly highlight differences in the results.

When performing clustering, the clustering labels do not tell you which class it is. If the clustering is successful, the data with each cluster label can be checked to see which class it belongs to. In the case of the low ARI scores pesented in 5, it would be fair to assume that the wrong cluster label has been compared to the the true labels, however by inverting the clustering results, the ARI score does not change. The ARI is adjusted for chance, and will penalize these clusters for being too large [7].

The results from SPCN were worse than expected. Based on the results of Roy et al. [24] it was expected to be the best performing stain normalization method. It is possible that a SPCN implementation optimized for PAS stained images would improve the results for SPCN. The results could also indicate that the method was not well represented in these experiments due

to the non-optimized parameters for clustering and dimensionality reduction.

The process of normalization, feature extraction, dimensionality reduction and clustering is time consuming. Each experiment takes around two hours of data processing using the hardware that was available. This process is excluding the time to adjust algorithms for different reference images and stain normalization techniques. A realistic number of experiments was chosen based on the processing time. It seems that it would be beneficial to perform more experiments in order to properly evaluate the methods.

Chapter 7

Conclusions

Overall, the results for all methods show improved clustering results over not using stain normalization, provided an appropriate reference image is used. Using the 'high', 'amyloidosis' and 'necrosis' reference image from 3.4, the clustering results were improved in all stain normalization methods.

The figure 5.1 shows an overview of the results. Both the choice of stain normalization method and reference image appears important for optimizing the unsupervised learning algorithm on this data. While normalization with most of the reference images improved the clustering result, the image of global sclerosis stands out as a bad reference image. There could be multiple reasons for this, one theory is that the image does not accurately represent the typical glomerulus contents. The scarring is replacing structures inside the glomerulus, these structures would contain different colors from the ones that are present in this instance of global sclerosis.

Using this exact dataset [4], along with the same parameters for the algorithm, the best option to improve clustering will be to use the Reinhard method along with the high intensity reference image. Using this method, the ARI score was increased from 0.48 to 0.66, an improvement of 37.5 percent.

Answering the question of whether stain normalization should be included in an unsupervised learning algorithm for automatic classification of glomeruli,

the results indicate that it should. All these stain normalization techniques show improvement with at least four out of the six reference images. The results indicate that both intensity and structure should be considered when choosing a reference image.

Future work for this subject could include: More experiments, this can be achieved by using more reference images for each stain normalization technique. Fine tuning clustering parameters for each stain normalization technique, this would likely improve performance and more strongly highlight true potential of each technique. Fine tuning the CNN for glomeruli images by further training the pre-trained model. In this case, it would be ideal to either use a different dataset for training, or by using augmented images extracted from the dataset.

Bibliography

[1] Digital Slide Archive. Histomicstk. url: https://github.com/DigitalSlideArchive/HistomicsTK, accessed: 19.05.2023.

[2] Wesam Ashour Barbakh, Ying Wu, and Colin Fyfe. *Review of Clustering Algorithms*, pages 7–28. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[3] Boris Bikbov, Caroline A Purcell, Andrew S Levey, Mari Smith, Amir Abdoli, Molla Abebe, Oladimeji M Adebayo, Mohsen Afarideh, Sanjay Kumar Agarwal, Marcela Agudelo-Botero, et al. Global, regional, and national burden of chronic kidney disease, 1990–2017: a systematic analysis for the global burden of disease study 2017. *The lancet*, 395(10225):709–733, 2020.

[4] Weis Cleo-Aron, Bindzus Jan Niklas, Jonas Voigt, Runz Marlen, Gaida Matthias M. Hertjens Svetlana, Popovic Zoran V., and Porubsky Stefan. Assessment of glomerular morphological patterns by deep learning algorithms. *Journal of Nephrology*, 35:417–427, 2022.

[5] Insight Software Consortium. Itkcolornormalization. url: https://github.com/InsightSoftwareConsortium/ITKColorNormalization, accessed: 05.06.2023.

[6] Mark Haas, Surya V. Seshan, Laura Barisoni, Kerstin Amann, Ingeborg M. Bajema, Jan Ulrich Becker, Kensuke Joh, Danica Ljubanovic, Ian S.D. Roberts, Joris J. Roelofs, Sanjeev Sethi, Caihong Zeng, and J. Charles Jennette. Consensus definitions for glomerular lesions by light and electron microscopy: recommendations from a working group

of the renal pathology society. *Kidney International*, 98(5):1120–1134, 2020.

[7] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2:193–218, 1985.

[8] IBM. Machine learning. url: https://www.ibm.com/topics/machine-learning, accessed: 19.05.2023.

[9] IBM. What is a neural network? url: https://www.ibm.com/topics/neural-networks, accessed: 31.05.2023.

[10] ImageNet. About imagenet. url: https://www.image-net.org/about.php, accessed: 07.06.2023.

[11] Lei Jiang, Wenkai Chen, Bao Dong, Ke Mei, Chuang Zhu, Jun Liu, Meishun Cai, Yu Yan, Gongwei Wang, Li Zuo, et al. A deep learning-based approach for glomeruli instance segmentation from multistained renal biopsy pathologic images. *The American Journal of Pathology*, 191(8):1431–1441, 2021.

[12] Neel Kanwal, Fernando Pérez-Bueno, Arne Schmidt, Kjersti Engan, and Rafael Molina. The devil is in the details: Whole slide image acquisition and processing for artifacts detection, color variation, and data augmentation: A review. *IEEE Access*, 10:58821–58844, 2022.

[13] Joonsang Lee, Elisa Warner, Salma Shaikhouni, Markus Bitzer, Matthias Kretzler, Debbie Gipson, Subramaniam Pennathur, Keith Bellovich, Zeenat Bhat, Crystal Gadegbeku, Susan Massengill, Kalyani Perumal, Jharna Saha, Yingbao Yang, Jinghui Luo, Xin Zhang, Laura Mariani, Jeffrey B. Hodgin, Arvind Rao, and the C-PROBE Study. Unsupervised machine learning for identifying important visual features through bag-of-words using histopathology. *Scientific Reports*, 12, 2022.

[14] Andrew S Levey and Josef Coresh. Chronic kidney disease. *The lancet*, 379(9811):165–180, 2012.

[15] M. Macenko, M. Niethammer, J.S. Marron, D. Borland, J.T. Woosley, Xiaojun Guan, C. Schmitt, and N.E. Thomas. A method for normalizing histology slides for quantitative analysis. In *2009 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, pages 1107–1110. IEEE, 2009.

[16] T Soni Madhulatha. An overview on clustering methods. *arXiv preprint arXiv:1205.1117*, 2012.

[17] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction, 2020.

[18] OECD.AI. Adjusted rand index (ari). url: https://oecd.ai/en/catalogue/metrics/adjusted-rand-index, accessed: 02.06.2023.

[19] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks, 2015.

[20] Martin R. Pollak, Susan E. Quaggin, Melanie P. Hoenig, and Lance D. Dworkin. The glomerulus: The sphere of influence. *Clinical Journal of the American Society of Nephrology*, 9(8):1461–1469, 2014.

[21] Goutham Ramakrishnan, Deepak Anand, and Amit Sethi. Fast gpu-enabled color normalization for digital pathology, 2019.

[22] Waseem Rawat and Zenghui Wang. Deep convolutional neural networks for image classification: A comprehensive review. *Neural computation*, 29(9):2352–2449, 2017.

[23] Peter Rossing, M. Luiza Caramori, Juliana C.N. Chan, Hiddo J.L. Heerspink, Clint Hurst, Kamlesh Khunti, Adrian Liew, Erin D. Michos, Sankar D. Navaneethan, Wasiu A. Olowu, Tami Sadusky, Nikhil Tandon, Katherine R. Tuttle, Christoph Wanner, Katy G. Wilkens, Sophia Zoungas, and Ian H. de Boer. Kdigo 2022 clinical practice guideline for diabetes management in chronic kidney disease. *Kidney International*, 102(5):1–127, 2022.

[24] Santanu Roy, Alok kumar Jain, Shyam Lal, and Jyoti Kini. A study about color normalization methods for histopathology images. *Micron*, 114:42–61, 2018.

[25] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[26] Noriaki Sato, Eiichiro Uchino, Ryosuke Kojima, Minoru Sakuragi, Shusuke Hiragi, Sachiko Minamiguchi, Hironori Haga, Hideki Yokoi,

Motoko Yanagita, and Yasushi Okuno. Evaluation of kidney histological images using unsupervised deep learning. *Kidney International Reports*, 6(9):2445–2454, 2021.

[27] Abhishek Vahadane, Tingying Peng, Amit Sethi, Shadi Albarqouni, Lichao Wang, Maximilian Baust, Katja Steiger, Anna Melissa Schlitter, Irene Esposito, and Nassir Navab. Structure-preserving color normalization and sparse stain separation for histological images. *IEEE Transactions on Medical Imaging*, 35(8):1962–1971, 2016.

[28] Raymond Vanholder, Lieven Annemans, Aminu K Bello, Boris Bikbov, Daniel Gallego, Ron T Gansevoort, Norbert Lameire, Valerie A Luyckx, Edita Noruisiene, Tom Oostrom, Christoph Wanner, and Fokko Wieringa. Fighting the unbearable lightness of neglecting kidney health: the decade of the kidney*. *Clinical Kidney Journal*, 14(7):1719–1730, 04 2021.

[29] Nicolas Wagner, Moritz Fuchs, Yuri Tolkach, and Anirban Mukhopadhyay. Federated stain normalization for computational pathology. *Medical Image Computing and Computer Assisted Intervention*, pages 14–23, 2022.

[30] Angela C Webster, Evi V Nagler, Rachael L Morton, and Philip Masson. Chronic kidney disease. *The Lancet*, 389(10075):1238–1252, 2017.

Appendix A - Reinhard stain normalization

In [ ]:
```
import histomicstk as htk
import numpy as np
import scipy as sp
import skimage.io
import skimage.measure
import skimage.color
import matplotlib.pyplot as plt
import os
import cv2
import numpy as np

from tqdm import tqdm
from skimage.metrics import structural_similarity
from skimage.transform import resize
from matplotlib import pylab as plt
from matplotlib.colors import ListedColormap
from histomicstk.preprocessing.color_normalization import reinhard
```

Functions

In [ ]:
```
def hsi_mask(image):
    #HSI threshold values for masking white pixels
    #Source: HistomicsTK/saliency/cellularity_detection_threshold
    hue_min, hue_max = 0, 1
    saturation_min, saturation_max = 0, 0.2
    intensity_min, intensity_max = 220, 255

    #Converting image from RGB to HSI
    image_hsi = htk.preprocessing.color_conversion.rgb_to_hsi(image)

    #Applying thresholds to HSI image to obtain white pixel masks for each component:
    hue_mask = cv2.inRange(image_hsi[:,:,0], hue_min, hue_max)
    sat_mask = cv2.inRange(image_hsi[:,:,1], saturation_min, saturation_max)
    int_mask = cv2.inRange(image_hsi[:,:,2], intensity_min, intensity_max)

    #Converting HSI mask components to boolean arrays
    hue_mask = (hue_mask/255).astype(bool)
    sat_mask = (sat_mask/255).astype(bool)
    int_mask = (int_mask/255).astype(bool)

    #Combining HSI components into one mask
    hsi_mask = (hue_mask == 1) &(sat_mask == 1) & (int_mask == 1)
    return hsi_mask

def Reinhard(image,image_mask,ref,ref_mask):
    #Calculating color mean and standard deviation for the reference image using image and mask
    refMu, refSgm = htk.preprocessing.color_conversion.lab_mean_std(ref, mask_out=ref_mask)
    #Normalizing image using the reference mean and std, source image and respective white pixel mask
    imReinhard = htk.preprocessing.color_normalization.reinhard(im_src=image,target_mu=refMu,target_sigma=refSg
    return imReinhard
```

Single image example

In [ ]:
```
refIm_path = 'path to reference image'
testIm_path = 'path to test image'
refIm = skimage.io.imread(refIm_path)[:,:,:3]              #Reading images in RGB with 3 color channels
testIm = skimage.io.imread(testIm_path)[:,:,:3]

ref_mask = hsi_mask(refIm)                                 #White pixel mask for reference image
image_mask = hsi_mask(testIm)                              #White pixel mask for source image
imReinhard = Reinhard(testIm, image_mask, refIm, ref_mask) #Normalizing image using reinhard

plt.figure(figsize=(10,15),layout='tight')
plt.subplot(3,2,1)
plt.title('Target image')
plt.axis('off')
plt.imshow(testIm)
plt.subplot(3,2,2)
plt.title('Mask for target image')
plt.axis('off')
plt.imshow(image_mask)
plt.subplot(3,2,5)
plt.title('Processed image')
```

```
plt.axis('off')
plt.imshow(imReinhard)
plt.subplot(3,2,3)
plt.title('Reference image')
plt.axis('off')
plt.imshow(refIm)
plt.subplot(3,2,4)
plt.title('Mask for reference image')
plt.axis('off')
plt.imshow(ref_mask)
plt.savefig('single_image_example.png')
plt.show()
```

Variables

In [ ]:
```
Original_path = 'Path to the original dataset'
OriginalData = os.listdir(Original_path)                        #List of classes in directory
glomeruli = os.listdir(Original_path+'/'+OriginalData[0])       #List of all original images
ClassesReinhard = os.listdir(Original_path)                     #Classes for reinhard normalized images
NormalizedReinhard = []                                         #Initializing array to store normalized images
original_images = []                                            #Initializing array to store original images
```

Normalize all images (Time demanding)

In [ ]:
```
for i in tqdm(range(len(OriginalData))):                                #tqdm makes the progress bar
    glomeruli = os.listdir(Original_path+'/'+OriginalData[i])           #List of all filenames in the class
    categoty_path = Original_path+'/'+OriginalData[i]                   #String to class directory
    NormalizedReinhard.append([])                                       #Class
    original_images.append([])                                          #Class

    for j in range(len(glomeruli)):
        glomerulus = skimage.io.imread(categoty_path+'/'+glomeruli[j])  #Reading one image
        original_images[i].append(glomerulus)                           #Appending the original image for la
        image_mask = hsi_mask(glomerulus)                               #Mask for source image
        imReinhard = Reinhard(glomerulus, image_mask, refIm, ref_mask)  #Normalize image

        NormalizedReinhard[i].append(imReinhard)                        #Storing normalized image in array
```

Making folder structure, saving normalized images, printing 10 samples from all classes

In [ ]:
```
save_dir = 'Path to the directory for normalized images'
os.chdir(save_dir)
for i in range(len(ClassesReinhard)):
    os.chdir(save_dir)
    print(ClassesReinhard[i]
    os.mkdir(ClassesReinhard[i])
    os.chdir(ClassesReinhard[i])
    filenames = os.listdir(Original_path+'/'+ClassesReinhard[i])
    for j in range(len(NormalizedReinhard[i])):
        filename = filenames[j]
        img = NormalizedReinhard[i][j]
        skimage.io.imsave(filename, img) #Saving the image in the correct folder

    glomeruli = os.listdir(Original_path+'/'+OriginalData[i])
    categoty_path = Original_path+'/'+OriginalData[i]

    print('Normalized row 1, Original row 2')
    plt.figure(figsize=(42,6), layout='tight')
    for k in range(9):
        plt.subplot(2,9,k+1)
        plt.imshow(NormalizedReinhard[i][k])
        plt.axis('off')

    plt.figure(figsize=(43,6), layout='tight')
    for s in range(9):
        glomerulus = skimage.io.imread(categoty_path+'/'+glomeruli[s])
        plt.subplot(2,9,s+10)
        plt.imshow(glomerulus)
        plt.axis('off')

    plt.show()
```

In [ ]:
```
org_dir = 'Path to original dataset'
prc_dir = 'Path to normalized dataset'
```

```
struc_sim = []         #Structural similarity index metric score list
ssim_trace = []        #Structural similarity index metric score array (in case we want to match to data)
os.chdir(save_dir)
for i in range(len(ClassesReinhard)):
    os.chdir(prc_dir)
    os.chdir(ClassesReinhard[i])
    filenames = os.listdir(org_dir+'/'+ClassesReinhard[i])
    ssim_trace.append([])
    for j in tqdm(range(len(NormalizedReinhard[i]))):
        filename = filenames[j]
        prc_img = NormalizedReinhard[i][j]
        org_img = original_images[i][j]

        struc_sim.append(structural_similarity(org_img, prc_img, channel_axis=2))
        ssim_trace.append(structural_similarity(org_img, prc_img, channel_axis=2))
```

In [ ]:

```
mean_ssim = np.mean(struc_sim)
print(mean_ssim) #The mean SSIM for the entire dataset
```

Appendix B - Clustering

Importing libraries

In [ ]:
```
import os
import tensorflow as tf
import numpy as np
import pandas as pd
import sklearn
import matplotlib.pyplot as plt
import cv2
import umap
import tqdm


from keras.applications.nasnet import NASNetLarge, preprocess_input
from sklearn.datasets import load_digits
from sklearn.preprocessing import StandardScaler
from tensorflow import keras
from sklearn.manifold import TSNE
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img
from tensorflow.keras.preprocessing import image
from tqdm import tqdm
from sklearn.mixture import GaussianMixture
from sklearn.cluster import KMeans
from sklearn.metrics.cluster import adjusted_rand_score
```

Choose experiment

In [ ]:
```
Macenko = ['Macenko_high_intensity', 'Macenko_low_intensity', 'Macenko_perfect_intensity', 'Macenko_Amyloidosis
Reinhard = ['Reinhard_high_intensity', 'Reinhard_low_intensity', 'Reinhard_perfect_intensity', 'Reinhard_Amyloi
SPCN = ['SPCN_high_intensity', 'SPCN_low_intensity', 'SPCN_perfect_intensity', 'SPCN_Amyloidosis', 'SPCN_Global_
Original = 'KlassifikationPorubsky_Anonymized'

choose_experiment = Original
filepath = 'path to folder containng all datasets' + choose_experiment
clustering_file = 'clustering_' + choose_experiment
feature_file = 'features_' + choose_experiment
figure_file = 'figures_' + choose_experiment
print('Clustering for images normalized with reference: ', choose_experiment)
print('Features will be saved at: ' + filepath + '/' + feature_file)
print('Figures will be saved at: ' + filepath + '/' + figure_file)
print('Clustering results will be saved at: ' + filepath + '/' + clustering_file)
```

Activate GPU

In [ ]:
```
physical_devices = tf.config.list_physical_devices('GPU')  #Fetch data about available hardware
print("Num GPUs Available: ",len(physical_devices))
os.environ['CUDA_VISIBLE_DEVICES']='0'                      #Selecting GPU to use
config=tf.compat.v1.ConfigProto()
session = tf.compat.v1.Session(config=config)
```

Load pre-trained CNN

In [ ]:
```
input_shape = (224, 224, 3)
base_model = tf.keras.applications.NASNetLarge(weights='imagenet',include_top=False, input_shape=input_shape) #I
base_model.trainable = False
out = tf.keras.layers.GlobalAveragePooling2D()(base_model.output)
model = keras.Model(inputs=base_model.input, outputs=out)
model.compile()
print(len(model.layers))
model.summary()
```


Setting parameters for UMAP and GMM

In [ ]:
```
#Clustering and umap parameters provided by Dr. Hrafn Weishaupt
clusters = 2
neighbors = 30
```

```
mDist = 0.1
spread = 9
local_con = 9
metric = 'manhattan'
init = 'spectral'

#Needed for feature extraction
filenames = os.listdir(filepath+'/'+'All_classes')
features = np.empty((len(filenames), 4032))
```

Feature extraction

In [ ]:
```
#Skip if loading features from file
for i in tqdm(range(len(filenames))): #tqdm gives loading bar
    img = load_img(filepath + '/All_classes/' + filenames[i].replace('"',''), target_size=(224,224)) #Load imag
    img_array = image.img_to_array(img) #Convert image to array
    img_tensor = np.expand_dims(img_array, axis=0) #Expand dims to get tensor
    input_img = tf.keras.applications.nasnet.preprocess_input(img_tensor, data_format=None) #feature extraction
    features[i,:] = (model(input_img)) #Store features in array

#To save features to file
os.chdir(filepath)
np.save(feature_file, features, allow_pickle=True)
```

Alternative, load features from file

In [ ]:
```
#How to load features
os.chdir(filepath)
features = np.load(feature_file+'.npy')
np.save(feature_file, features, allow_pickle=True)
```

Dimensionality reduction and clustering

In [ ]:
```
#Making the UMAP dimensionality reducer
reducer_umap = umap.UMAP(n_neighbors=neighbors, min_dist = mDist, spread = spread, init = init,
                         local_connectivity = local_con, metric = metric,n_components=clusters) #Initialize dime
DimRed_features = reducer_umap.fit_transform(features) #Fit the model to data

#Fit a gaussian mixture model to the data
Clustering_model = GaussianMixture(n_components=clusters, n_init=100, init_params = 'k-means++').fit(DimRed_fea

#Using the clustering model to predict on the data
clustered_data = Clustering_model.predict(DimRed_features)
```

True labels for global sclerosis and Adjusted Rand Index

In [ ]:
```
TrueLabels = []
all_labels = []
for i in range(len(filenames)):
    cl = filenames[i].split('_')[1]
    if (cl == 'Global-Sclerosis'):
        TrueLabels.append(1)
    else:
        TrueLabels.append(0)
    temp = (filenames[i].split('_')[0])
    all_labels.append(int(temp.split('n')[1]))

ARI = adjusted_rand_score(TrueLabels, clustered_data)
print('Adjusted Rand Index result: ' + str(ARI))
```

Clustering results, filenames and true labels saved to file

In [ ]:
```
d = {'Filenames': filenames, 'Clustering results': clustered_data, 'True labels': TrueLabels}
df = pd.DataFrame(data=d) #Puts cluster labels and true labels in dataframe
os.chdir(filepath)
df.to_csv(clustering_file+'.csv') #save dataframe to file
```

In [ ]:
```
cluster_res = df['Clustering results']
true_labels = df['True labels']
CS = []
TL = []
for i in range(len(cluster_res)):
```

```
    CS.append(cluster_res[i])

    TL.append(true_labels[i])
```

Visualization of clustering and true labels

In [ ]:
```
plt.figure(figsize=(10,6))
plt.subplot(1,2,1)
plt.title('Clustering results')
plt.scatter(DimRed_features[:,0], DimRed_features[:,1], c=CS, cmap='Accent', s=5)
plt.axis('off')
plt.subplot(1,2,2)
plt.title('True labels')
plt.scatter(DimRed_features[:,0], DimRed_features[:,1], c=TL, cmap='Spectral', s=5)
plt.axis('off')
plt.savefig(figure_file+'.png')
plt.show()
```