Classification and Image Segmentation of Pollen Grains

Alexander Crompton

Masteruppsats i matematisk statistik
Master Thesis in Mathematical Statistics

Mathematical Statistics
Stockholm University
Master Thesis **2023:17**
http://www.math.su.se

Classification and Image Segmentation of Pollen Grains

Alexander Crompton[*]

September 2023

Abstract

This thesis presents an automated approach for the classification of Swedish pollen grains. The research aims to enhance the accuracy of pollen classification while automating the process of identifying pollen. The proposed methodology involves pre-processing, segmentation using clustering, and a convolutional neural network, classifying eight different pollen types. In the pre-processing phase methods are compared to find the best method of reducing dirt and other noise found in the microscopic images of the pollen preparations. The segmentation of individual pollen grains is done using the k-means clustering algorithm, which performance extraction and identification of pollen grains compared to the watershed method in terms of computational efficiency. For pollen classification, six convolutional neural networks were trained their performance were compared using loss and accuracy. The models were trained using RGB and grey versions of 3 sets of images of the same segmented pollen grains. Results showed using a mixture of images was the most accurate and achieved even higher accuracy when using the maximum softmax probability over multiple image depths to predict pollen type. To achieve full automation for pollen classification wider sets of data with more variation is needed for the convolutional neural networks along with further research into segmentation of pollen.

[*]Postal address: Mathematical Statistics, Stockholm University, SE-106 91, Sweden. E-mail: alexcrompton@hotmail.se. Supervisor: Martin Sköld.

Acknowledgements

I would like to express my deepest gratitude to my supervisor Martin Sköld for the idea of this thesis and for all your help and guidance throughout the entire work. It has been a true pleasure working with someone with your knowledge and enthusiasm.
I would also like to give a special thanks to Björn Gedda for scanning all pollen and also taking his time to manually classify pollen grains. Without these contributions this work would not have been possible.
.

Contents

1 Introduction

For many of us the most well-known effects of pollen, is its role in triggering allergic reactions in the spring time. It is a common allergen that can cause hay fever and other respiratory allergies [1]. When inhaled, pollen grains can irritate the nasal passages and trigger allergic symptoms such as sneezing and itchy eyes. Understanding the types of pollen that are prevalent in different regions, their seasonal patterns, and their interactions with the human immune system is crucial for managing and mitigating allergic reactions, making pollen an important topic of research in allergy and immunology.

Pollen also has practical implications in agriculture and food production. Pollination, the process by which pollen is transferred from the male to the female reproductive organs of flowers, is essential for the production of fruits, vegetables, and seeds. Studying pollen and its interactions with pollinators, such as bees and butterflies can help improve crop yields, optimise agricultural practices, and ensure food security.

Due to its biological impact, pollen are also of high interest in image recognition tasks. Pollen grains exhibit diversity in shape, size, and surface ornamentation, which can make them visually distinct and recognisable under a microscope. This unique visual characteristic of pollen grains opens up possibilities for using image recognition techniques to identify and classify different types of pollen. For instance, pollen recognition can be applied in ecological studies to track changes in plant populations, identify plant species in environmental samples, and monitor ecosystem health. In recent years there have been many studies about classifying different pollen species especially using modern advanced neural networks.

The Swedish Natural History Museum in Stockholm is responsible for all the counting and forecasting of pollen in Sweden. This is done by using pollen traps placed in different parts of Sweden and then manually counting the amount of each species. This can be tedious and time-consuming. In this thesis we will look into methods in machine learning to start to automate this process.

The purpose of this thesis is to research a general method for automation of classifying pollen. A comprehensive view of the entire process from segmentation of specific pollen grains to classifying the grains will be reviewed. A comparison to previous work and their potential solutions to common issues for real world deployment of automatic classification will be discussed.

Since a large portion of previous research within automatic pollen grain analysis has been specifically for classification on available data sets, a large portion of this thesis will be comparing segmentation methods and preprocessing of the microscopic images. One research article that discusses the segmentation of pollen grains is "Pollen segmentation and feature evaluation for automatic classification in bright-field microscopy" by Redondo

et al. [5]. While the article does discuss many practical solutions to segmentation, the segmentation mask is done manually. In this thesis we aim to find ways to automatically extract pollen. For this pre-processing and segmentation of the pollen images, Supervised and Unsupervised learning techniques will be compared. For the classification Deep Learning Neural Networks will be applied. With high performing classification already being achieved in previous research, the focus of the classification task will mainly lay on comparing how the performance of the Deep Learning Models holds up in comparison using different image focus, grey images and Image Augmentation.
In recent years, the field of pollen analysis has seen a growing interest in developing methodologies for machine learning-based classification of pollen grains. Previous research have contributed to this area, shedding light on various aspects of the process. One such is the overview by Viertel and König [2] focused on pattern recognition methodologies for pollen grain image classification. Viertel and König explored different approaches for the classification of pollen grains based on their visual patterns. Their work provided valuable insights into the use of pattern recognition techniques, highlighting the potential for accurate and efficient classification.
Specifically, we will delve into the workings of these approaches, analyse their performance, and explore additional aspects such as image pre-processing and augmentation techniques prior to Convolutional Neural Network (CNN) based classification. Another article providing insights to the big potential in CNNs, is the research article by Sevillano et al.[6]. The resulting CNN model classifies 46 different types of pollen to 0.98 accuracy, demonstrating higher accuracy than expert palynologists.
By examining and drawing upon the advancements made in the field of pollen analysis and machine learning classification, the aim is to contribute to the development of efficient and accurate methodologies for pollen classification.
In the upcoming chapters of this paper, we provide a structured overview of our research. The ”Materials” section outlines the essential information on data, types of images and types of pollen grains that were used in the thesis. In the ”Materials' Background” section, the reader is offered insight into the statistical methods used along with related work. The subsequent ”Methods” section details the systematic process employed for analysis. The following ”Results” section, unveils the outcomes of our research followed by the ”Conclusion” which summarises our findings. Lastly the ”Discussion” section discusses results and potential for future research in pollen automation.

2 Materials

In the Materials Section, a detailed description of the data that has been used in this study along with the biological background is provided. The text will inform what material the work has been done on and how these pollen images have been prepared. Factual details about the pollen and the images used in this study will be provided.

2.1 Pollen data

Pollen grains vary in size, shape, and texture, for instance, some of the pollen grains are smooth, while others have a rough surface. Additionally, the size of the pollen grains can also vary, with some being much larger than others. To gain basic understanding in how to identify specific pollen types the "Manual for Pollen Analysts" by Britt Berggren [3] was used. The manual provides comprehensive guidance for pollen analysts, covers various aspects of pollen analysis and microscopic identification of pollen grains.
There will be a focus on pollen often found in Sweden with the reason being that we wanted to direct our attention to develop automation for pollen often found in Sweden. In addition to that the study lays a focus on pollen that are spherical or uniform in shape. Examples of pollen that are not included in this are the Gymnosperms, Picea (spruce) and Pinus (pine) which have "air bubbles" on either side of the main body meaning they are easier to separate from other pollen types by eye, but also more difficult to automate.

2.1.1 Pollen preparations

Reference preparations were created by manually dipping a small clump of coloured gelatine into a sample of known pollen types onto a microscopic slide. This process ensured that the reference preparations contained a the specific variety of pollen we were looking to classify.
A full microscopic slide of pollen is seen in Figure 1. To get a detailed view of the pollen grains the slide is divided into smaller square segments, systematically from top left to bottom right into and images are then taken. The image segment can range from containing no pollen at all to more than 30 particles. The grains can lay on any place of an image and are often cut in half between the different image segments and may therefore not be possible to classify. For the training data we are provided with images that contain a specific species of pollen. The first challenge is to segment individual grains of each pollen type before training a neural network to classify between them. A general inconvenience along with the cut pollen grains, for segmenting individual pollen from the larger microscopic image is that image slices often contain foreign particles along with pollen. In Figure 2 some examples how the image segments look like along with the properties

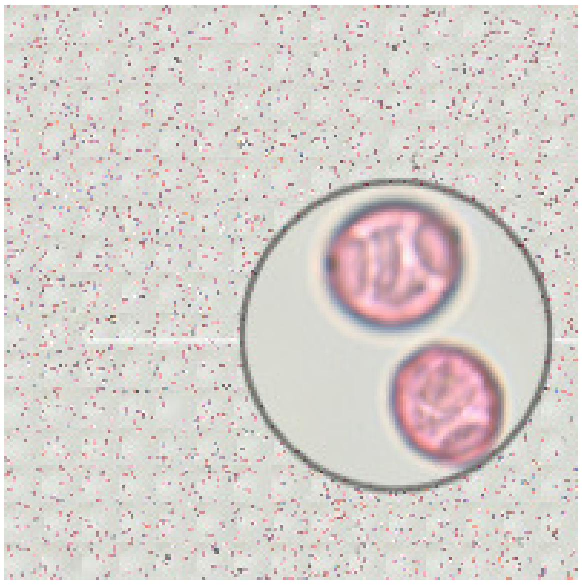


Figure 1: An example of a full microscopic slide containing pollen, with a zoomed image of pollen grains to illustrate the contents.

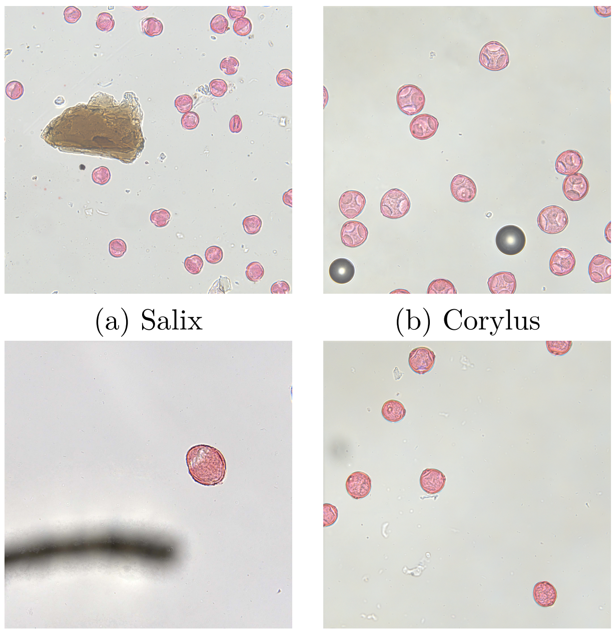


Figure 2: A square grid of four example pollen images: (a) Salix, (b) Corylus, (c) Quercus, (d) Betula.

and issues that needed to be addressed before segmentation.
Preparations may differ in staining colour and background and we want to avoid our networks overfitting based on factors that may vary between preparations. Images of mixture preparations of pollen are used in this study to test the quality of the Network on a new set of images with a mixture of pollen types. Since all the pollen on the mixed microscopic slide have the same background and staining, these grains can be used to test the quality of the network's predictions on the different pollen species. The mixture preparations consist of a combination of all pollen types the neural network was trained on excluding Rumex.

2.1.2 Image details

The images were captured using a Teledyne Luminera Infinite 3 camera.
The scanning of the reference preparations was predominantly performed directly to JPEG format. Some of the earliest preparations were converted from BMP to JPEG.
For scanning, a 40x objective lens was used in conjunction with a 6-megapixel camera. Determining the exact magnification is challenging as it depends on the pixel sizes and the extent of image enlargement. However, when using a 40x objective lens, a 10x eyepiece provides a magnification of 400x, which can be considered the equivalent magnification.
To enhance visibility, the pollen grains were stained with Safranin. This staining method was chosen to ensure clear visualisation of the pollen grains and is an important consideration since there are other dyes available for this purpose. An important aspect of the Safranin is that it does not colour other debris that are found on the microscopic slide. Note that the colour of the stained pollen fades over time which could lead to high variation in colour between different preparations.
The photographs are taken at different depths, in addition images the microscope software generates a compilation image focused for all depths. The compiled image provides a comprehensive view of the specimen under investigation, revealing details that may not be apparent in a single focal plane. However this is not to say a single focus plane could be equally or more revealing to identify type of pollen. Note that the orientation of the grains in the images can vary, as pollen grains can lay in different ways on the microscope slide. The raw data consists of separated samples of each type of pollen in different fields of focus along with the compiled image. In the following sections we will present and illustrate examples of how these images look like, explore the images and present the data for each species.

2.1.3 Image data

Each microscopic slide had a different number of photos taken at microscopic depths, the microscopic slides were of different sizes meaning some pollen have more images. In Table 1 the number of slices which indicates how many depths the pollen have been photographed and number of images segments are included. Also the total number of pollen that were segmented from each slice is provided, this number is the same for all slices since segmentation is done on the compiled image and then cut on the same place for each image. An important note is that extracted pollen is not the real number of pollen grains in the microscopic slide, since there might be pollen that are not segmented and some pollen grains are cut in half between the images.

Table 1: Data Description, Slices: Number of depths, Image Segments: Number of segments the microscopic slide was divided into, Extracted pollen: Total number of different pollen grains that were extracted

Pollen Type	**Slices**	**Image Segments**	**Extracted Pollen**
Ulmus	14	594	4966
Salix	14	373	5995
Rumex	15	210	1009
Quercus	15	450	2283
Fraxinus	15	374	4216
Corylus	13	225	2665
Betula	13	225	1954
Alnus	28	225	553

2.2 Image examples

In the following sections, visual examples of the eight pollen types after segmentation are provided for the compiled images and of images with a single slice.

2.2.1 Compiled images

The images in Figure 3 are examples of pollen data that we have collected after segmentation. Each image represents a different type of pollen and are compiled versions of the images of all depths of focus.

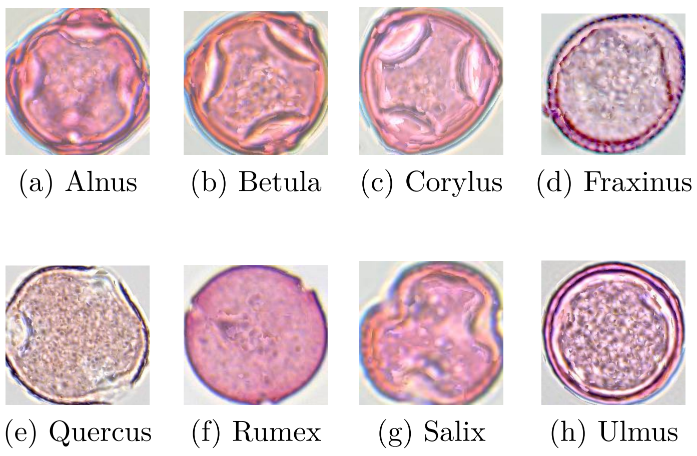


Figure 3: Eight compiled images of different pollen types: (a) Alnus, (b) Betula, (c) Corylus, (d) Fraxinus, (e) Quercus, (f) Rumex, (g) Salix, (h) Ulmus.

2.2.2 Sharpest slice images

Figure 4 are some new examples of single pollen grains after segmentation and determining the sharpest version for the specific image segment. How the sharpest images were found can be seen in the method section. Compared to the compiled images the sharpest image slices generally have a clearer visual texture of the surface of the grain, while the compiled image often highlight the important features of the pollen such as the edges.

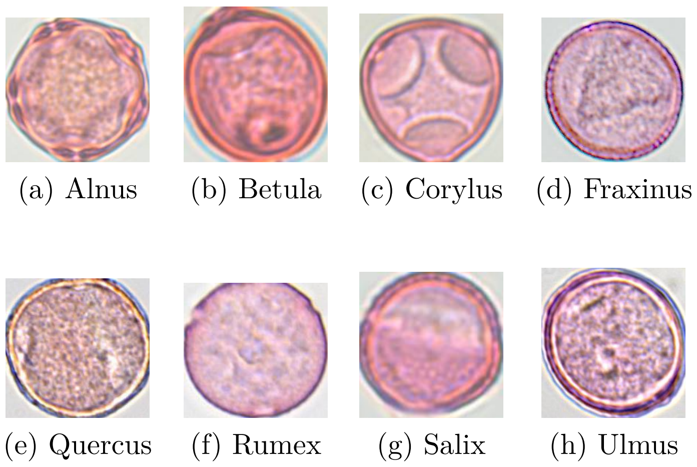


Figure 4: Eight sharp Single-Slice images of different pollen types: (a) Alnus, (b) Betula, (c) Corylus, (d) Fraxinus, (e) Quercus, (f) Rumex, (g) Salix, (h) Ulmus.

3 Method's background

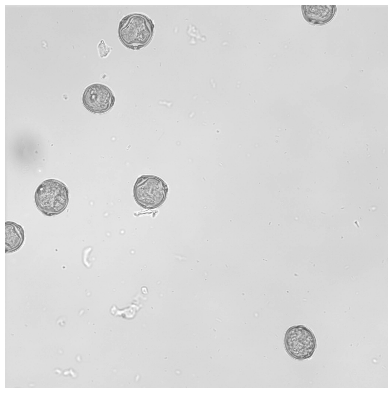
The method's background provides the reader with an overview of the statistical methods relevant to pollen, segmentation and classification while also taking a look at previous research on the topic. Understanding the statistical techniques is crucial for comprehending the subsequent discussions and analysis presented in the methods used and presented in the results in this paper.

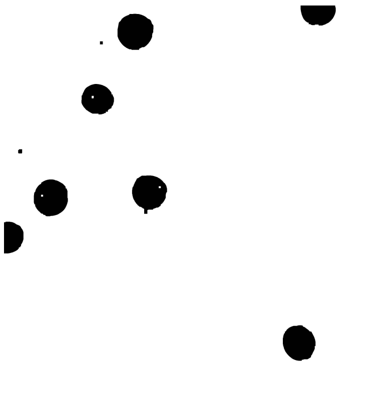
3.1 Related work

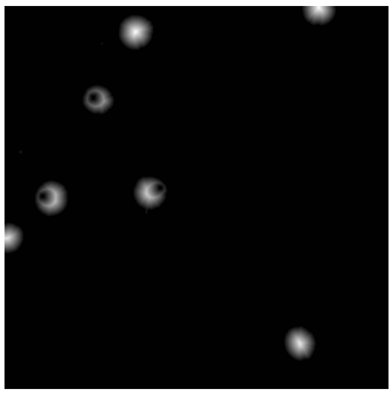
With pollen analysis being an important task in ecological research, several papers discussing methodologies for pollen grain classification have been published in recent years. One of such being the overview article by Viertel and König. [2] conducted a study of pattern recognition methodologies for pollen grain image classification. However as discussed in Olsson et al. [4] there are few papers that offer insight to the entire process of pollen analysis for real world classification. In the paper they proposed an efficient and robust approach for pollen analysis using deep learning. They performed pollen grain segmentation using gradient masks and dilation with linear structuring elements. They then used a CNN for pollen grain classification based on the segmented images. Suggested work includes looking at preprocessing of images and Augmentation of the images before CNN.
In the research article by Redondo et al. [5] pollen segmentation and methods for automatic classification were discussed. While the segmentation in the article was done by manually cropping pollen there were still details and discussion useful in this work, especially when it came to post-processing to improve the segmented regions.
In an article by Sevillano et al.[6] the authors demonstrated the incredible potential for precise automatic classification of different pollen types using CNNs. It's noteworthy to mention that their images are from dark field microscope in contrast to ours, and their dataset obviously features a significantly larger number of classes.
In their study titled "Automatic Pollen Classification and Segmentation Using U-Nets and Synthetic Data," Boldeanu et al. [6] present an innovative approach for pollen classification and segmentation utilising U-Net architectures. Their work highlights the potential of deep learning techniques and synthetic data generation in the context of pollen analysis. However for the purpose of this thesis, manually creating masks separating different pollen types and junk was not a possibility due to the expertise manually classifying pollen demands.
These proposed methodologies provided valuable insights, in the following sections we will take a further look at the methods in the segmentation and classification of the pollen grains.

3.1.1 Segmentation

The approach to Olsson et al. [4] segmentation process is discussed in the following section. In Figure 5 a visualisation of the segmentation described by Olsson is shown. Note that In Olsson's paper the Segmentation process is done using MATLAB's Image Processing Toolbox. First, the greyscale image (a) was thresholded (b) to obtain a binary image. Then, a distance transform was applied to the thresholded image (c) to compute the distance from each pixel to the nearest object boundary. Finally, a watershed algorithm was used to generate a watershed mask (d) based on the distance transformed image (c). These individual segmented Pollen in the watershed mask are then extracted through small bounding boxes.


((a)) greyscale Image


((b)) Thresholded Image


((c)) Distance Transformed Image

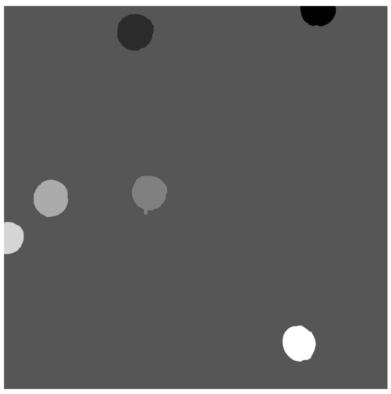

((d)) Watershed Mask

Figure 5: Segmentation Process of Betula in Figure 2d

Although the methodology in Figure 5 does a fine job in extracting pollen grains, it is not flawless. The chosen example image, Betula (d) in Figure 2 is clean and only contains pollen grains and the resulting watershed segments 6 out of the 7 grains. However as discussed, each image slice may contain

particles or objects that are not pollen, and this segmentation process does not work to separate the pollen from these foreign particles. This means the segmentation will work on all objects that are seen in the image, including the foreign objects.
A proposition to this limitation is to perform pre-processing of the pollen image before thresholding the grey image. The idea behind this is to highlight the pollen grains in the grey image so that the threshold image will only contain the pollen grains.To execute this we will look at a few different proposed methods and compare the threshold images. In Redondo et al. [5] hole filling on the segmented images is performed to ensure the integrity and continuity of segmented regions by addressing any interior holes. They also execute an opening operation which is conducted using a 15x15 kernel, combining erosion and dilation steps. This procedure refines the segmentation outcome, removing residuals and enhancing the quality of the segmented regions.

3.1.2 Classification

A widely used method for image classification, also used and presented in both Viertel and König. [2] and Olsson et al. [4] with high precision is CNNs. In [4] it is specifically discussed that the Deep learning method is favoured due to the quality of results, especially when dealing with a large amount of data to train the model. For some researchers the accuracy for the Deep learning model have been found to be close to nearly perfect with Precision of 0.98 and sometimes even higher. However even if the pollen grains can be classified with high accuracy there are still many many problems that need to be researched. Difficulties to create a benchmark set to validate methods is described to be a problem, due to different image qualities and colouring of pollen. Since pollen is often scanned through different microscopes and stained using different methods, images of the same grain can vary greatly. A real world challenge is to be able to accurately classify grains in different conditions.
In the research article ”Precise automatic classification of 46 different pollen types with convolutional neural networks” by Sevillano et al.[6] data augmentation is done on the images before feeding them into the neural network to handle the different conditions. The augmentation is said to ”provide a wider variety of spatial features” which is done by procedures such as rotating, cropping and reflecting. The dark field microscope images clearly provide different conditions for classification to our microscopic images, visually the features such as texture and edges are more visible. Instead comparing a model with grey images could be interesting for our case.

3.2 Image processing

In the following section the theory for the image processing used is provided. The theory in this part for PCA and LDA is provided by the book “The Elements of Statistical Learning”. [7].

3.2.1 Principal component analysis

Principal Component Analysis (PCA) is an unsupervised learning technique used for dimensionality reduction. It is a way of finding the directions in which the data varies the most, and projecting the data onto those directions to obtain a lower-dimensional representation that captures most of the variance in the original data. The purpose of using PCA would be to find a principle component acting as a manifold to separate the important features of the pollen images, when creating a greyscale image.
Let X be an $n \times 3$ matrix, where each row represents the RGB colour values of a pixel in the image, and let $\mathbf{x}_i$ be the RGB colour vector of the i:th pixel. We seek to find a one-dimensional representation of the colour information of the image, which can be used to create a greyscale version of the image. PCA can be performed on the matrix X to find the principal components of the colour data. The covariance matrix of X can be denoted by S, which is given by:

$$S = \frac{1}{n} X^T X$$

where X^T is the transpose of X. The eigenvectors of S represent the directions of maximal variance in the colour data, and the corresponding eigenvalues represent the amount of variance explained by each direction.
Let $\mathbf{u}_1$ be the eigenvector of S corresponding to the largest eigenvalue, which represents the direction of maximal variance in the colour data. Each colour vector $\mathbf{x}_i$ can be projected onto the one-dimensional manifold defined by $\mathbf{u}_1$ by computing the dot product:

$$z_i = \mathbf{x}_i^T \mathbf{u}_1$$

The resulting vector $\mathbf{z}$ represents the one-dimensional representation of the colour information of the image.
To create a greyscale version of the image, the values of $\mathbf{z}$ can be mapped to the range $[0, 1]$. This can be done by normalising the values of $\mathbf{z}$ using the formula:

$$y_i = \frac{z_i - \min(\mathbf{z})}{\max(\mathbf{z}) - \min(\mathbf{z})}$$

where y_i represents the greyscale value of the i:th pixel in the image. This normalisation ensures that the greyscale values are in the range $[0, 1]$.

The vector $\mathbf{y}$ can be reshaped into an $n \times 1$ matrix Y, and the columns of Y can be stacked to create a greyscale image matrix G. The resulting matrix G represents the greyscale version of the original image.

3.2.2 Linear discriminant analysis

Linear Discriminant Analysis (LDA) is a supervised learning algorithm used for classification problems. It is particularly useful when the classes in the image are well-separated and the number of features is large relative to the number of pixels. LDA is based on the idea of finding a linear combination of the features that best separates the classes.
In the problem of separating pixels in an image into different classes based on a given mask, we can define a binary response vector $y \in 0, 1^n$ that contains the class labels for all pixels. Let $y_i \in 0, 1$ denote the class label for the i-th pixel, where $y_i = 1$ if the pixel belongs to the desired class, and $y_i = 0$ otherwise. To do this we manually create training data by drawing out a mask, marking each pollen on the image.
The goal of LDA is to find a separating plane that maximises the separation between the pollen pixels and the background pixels. We can define this plane using a linear function of the feature vector x_i:

$$f(x_i) = \beta^T x_i \tag{1}$$

where $\beta \in R^3$ is the vector of coefficients for the linear function.
To learn the coefficients β, we need to maximise the following criterion:

$$J(\beta) = \frac{\beta^T S_B \beta}{\beta^T S_W \beta} \tag{2}$$

where S_B is the between-class scatter matrix and S_W is the within-class scatter matrix. These matrices can be defined as:

$$S_B = \sum_{i=0}^{1} N_i(\bar{x}_i - \bar{x})(\bar{x}_i - \bar{x})^T \tag{3}$$

$$S_W = \sum_{i=0}^{1} \sum_{j:y_j=i} (x_j - \bar{x}_i)(x_j - \bar{x}_i)^T \tag{4}$$

N_i represents the count of pixels in each class $i = 0, 1$. For S_B calculate the overall centroid $\bar{x}$ of all the feature vectors in the dataset. For each class, calculate the difference between the class centroid $\bar{x}_i$ and the overall centroid $\bar{x}$, and compute the outer product of this difference with itself.
For S_W calculate the centroid $\bar{x}_i$ for background and pollen, then calculate the deviation of each pixel's feature vector x_j from its class centroid $\bar{x}_i$, and compute the outer product of this deviation with itself.

Maximising $J(\beta)$ with respect to β leads to the following solution:

$$\hat{\beta} = S_W^{-1}(\bar{x}_1 - \bar{x}_0) \tag{5}$$

Once we have learned the coefficients $\hat{\beta}$, we can use them to compute the distance of each pixel from the separating plane:

$$d_i = f(x_i) = \hat{\beta}^T x_i \tag{6}$$

We can then use these distances to create a greyscale image that highlights the regions of the original image that correspond to the desired class. Specifically, we can assign each pixel a greyscale value based on its distance from the separating plane. To get the grey image the following linear mapping is used:

$$\text{greyscale value} = \begin{cases} 0 & \text{if } d_i < 0 \\ 255 & \text{if } d_i > 1 \\ 255 \cdot d_i & \text{otherwise} \end{cases} \tag{7}$$

By finding a separating plane that maximises the separation between the two classes, LDA highlights the regions of the original image that correspond to the desired class. Since LDA is a supervised learning method, the purpose will be to find a separating plane using training images, before using this plane to create grey versions of all images.

3.3 Segmentation theory

In the following section the theory for the segmentation algorithms are provided. The purpose of the segmentation algorithm's in pollen segmentation is to separate the connected components of the foreground of the image. In our case, we want to separate pollen grains that are connected or lay on top or beneath other grains. We will compare the performance of the K-means algorithm with the Watershed algorithm which is used in previous research.

3.3.1 Watershed algorithm

The Watershed algorithm is described by Gonzalez and Woods [8] in detail, in this section the the basic interpretations are provided, a more detailed description is provided in the book. Watershed is a mathematical image segmentation technique, dividing an image into distinct regions based on a priority map. The idea behind the watershed algorithm is to treat the image as a topographical surface, where each pixel has a "height" value. The Algorithm then identifies marked areas specified representing boundaries in the image. The markers can be specified by labelling based on connected components. The algorithm starts by identifying potential markers, which are located at the local maxima of the intensity values in the greyscale

version of the image. These markers are used as starting points for the watershed algorithm to segment the image. As a pre-processing step in the watershed segmentation calculates the distance of each pixel in the image to the nearest marker. It assigns a value to each pixel based on its distance from the marker, where pixels closer to the marker have lower values, and pixels farther away have higher values. This distance map is then used to guide the watershed algorithm.
Foreground (FG) and background (BG) are two important concepts in watershed segmentation. The markers that are initially placed on the image represent FG and BG regions. FG markers are used to indicate pixels that are definite pollen grains, while BG markers represent regions that are known to be the background or regions to be excluded from segmentation. The values in the distance map are then used on the FG and BG of the image to separate objects and mark regions to segment.

3.3.2 K-Means

The theory for K-means is provided by the book "The Elements of Statistical Learning" [7]. K-means is a clustering algorithm used in machine learning to identify clusters of similar data points in a given dataset.
With a set of n data points with 2-dimensional coordinates (image pixel location) $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$ and a predefined number of clusters K, the K-means algorithm partitions the data points into K clusters $C_1, C_2, ..., C_K$, minimising the within-cluster sum of squares (WCSS) . WCSS is defined as the sum of the squared Euclidean distances between each data point and its assigned cluster centroid:

$$WCSS = \sum_{i=1}^{K} \sum_{(x,y) \in C_i} [(x - \mu_{x,i})^2 + (y - \mu_{y,i})^2] \tag{8}$$

where $\mu_{x,i}$ and $\mu_{y,i}$ are the x and y coordinates of the centroid of cluster C_i, respectively. The K-means algorithm looks to find the best fitting cluster centroids $\mu_{x,1}, \mu_{y,1}, \mu_{x,2}, \mu_{y,2}, ..., \mu_{x,K}, \mu_{y,K}$ that minimise WCSS.
A step-by-step description of the Algorithm:

1. Choose the number of clusters K that you want to identify in the dataset.
2. Initialise K centroids randomly. These will be the initial cluster centers.
3. Assign each data point to the nearest centroid based on Euclidean distance.
4. Update the cluster centroids, computing the mean of all the data points assigned to each cluster.

5. Steps 3 and 4 are repeated until convergence, which is achieved when the cluster assignments no longer changes.

Since the assignment of the centroids in randomly assigned in the second step, the centers may not be the same, with different initialisations. Another limitation is that k-means assumes data is spherical, however since this paper limits pollen types to circular pollen this is ideal.

3.3.3 Gap statistic

When classifying on the validation dataset we can easily select the correct number of clusters (pollen) seen on the image. However since the extraction of pollen for the training data is done on an exceptionally large number of images, a method to find the ideal number of clusters needs to be "automated". One proposed method is discussed is to use the gap statistic. The gap statistic is a method for estimating the optimal number of clusters in a dataset and was introduced in Tibshirani et al. [9] . It works by comparing the within-cluster dispersion of a clustering solution to that of a null reference distribution. The idea is that if the within-cluster dispersion of the clustering solution is much smaller than that of the null distribution, then the clustering solution is a good fit for the data, and the number of clusters in the solution is optimal.
Letting X be a dataset with n observations in p dimensions, and let K be the number of clusters to be considered. Let $W(K)$ be the within-cluster dispersion for a clustering solution with K clusters, and let $W^*(K)$ be the corresponding dispersion for a null reference distribution.
The gap statistic is then defined as:

$$Gap(K) = \log(W^*(K)) - \log(W(K)) \tag{9}$$

The null reference distribution is generated by simulating B sets of data from a reference distribution with the same marginal distributions as X, but with no correlation between the variables.
The within-cluster dispersion $W(K)$ is calculated for the clustering solution with K clusters. Which is done using a the k-means clustering. The within-cluster dispersion is a measure of the total within-cluster variation of the data, and can be calculated as the sum of the squared distances between each observation and the center of its assigned cluster.
The dispersion $W^*(K)$ for the null reference distribution is calculated by applying the same clustering algorithm to each of the B sets of simulated data, and taking the average within-cluster dispersion over all B sets.
The optimal number of clusters K^* is then chosen as the value of K that maximises the gap statistic:

$$K^* = \arg, \max_K; Gap(K) \tag{10}$$

The reasoning behind the gap statistic is that if the within-cluster dispersion for a clustering solution with K clusters is much smaller than that of the null reference distribution, then the clustering solution is likely overfitting the data. On the other hand, if the within-cluster dispersion is similar to that of the null reference distribution, then the clustering solution is not capturing any meaningful structure in the data. The gap statistic provides a way to balance these two considerations and select the optimal number of clusters.

3.4 Artificial neural networks

This overview provides a concise introduction to the fundamental concepts of artificial neural networks as described by Ian Goodfellow's book 'Deep Learning' [10]. Topics covered include neurons, activation functions, weight and bias updates using gradient descent, as well as techniques such as momentum and adaptive learning rates, before we dive into CNN:s which is the method that will be used to solve the image classification task in the paper. This is done since it is important to first understand the concept of Artificial Neural Networks (ANN:s), of which CNN:s are a specific type.

ANN:s are a type of machine learning model inspired by biological neural networks, which are composed of interconnected neurons in the brain. ANN:s consist of interconnected nodes, also known as neurons, that are organised into different layers. Each neuron receives input signals from other neurons or from the input data, processes the input using a non-linear activation function, and produces an output signal, which is passed to other neurons in the network or to the output layer.

The input to an ANN is denoted by $\mathbf{x}$, and the output by $\mathbf{y}$. The goal of an ANN is to learn a function $f(\mathbf{x})$ that maps input data to an output that minimises a cost function $J(\theta)$. During training, the ANN adjusts its parameters, denoted by θ, to minimise the difference between the predicted output $\hat{\mathbf{y}}$ and the true output $\mathbf{y}$ associated with the input data.

ANN:s are commonly used for supervised learning tasks, where the training data consists of labelled examples, i.e., input data along with their corresponding output values. This allows the ANN to learn to associate input patterns with their corresponding outputs, and to generalise to new input patterns.

Once an ANN has been trained on labeled data, it can be used to make predictions on new, unlabelled data. This is done by applying the learned function to the input data to produce an output.

The simplest form of an ANN is the single-layer perceptron, which consists of a single layer of neurons that compute a linear combination of the input features and produce a binary output. However, ANN:s with multiple layers, called deep neural networks, can learn complex functions and can achieve state of the art performances on a wide range of tasks, such as

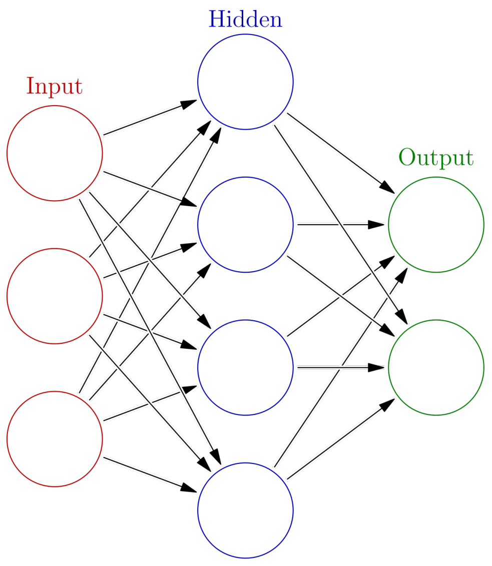


Figure 6: Simple Feed-Forward Neural Network Structure. Taken from [16]

image recognition discussed in this paper.
In Figure 6 we can see the structure of a simple Feed-Forward Neural Network composed of an input layer, one hidden layers, and an output layer. The input layer receives the input data, while the hidden layers perform feature extraction and nonlinear transformations on the input data using activation functions. The output layer produces the final output of the network.
CNN:s are a type of ANN that have been particularly successful in image recognition tasks. They consist of multiple layers of neurons that perform convolutions and pooling operations on the input data to extract features, followed by one or more fully connected layers that produce the final output. By using convolutional layers, CNNs can learn local and translation-invariant features in images, and by using pooling layers, they can reduce the dimensionality of the input data and make the network more computationally efficient. Before looking further into the detail of the convolution and pooling steps, we take a further look at the Architecture of the neurons in the ANN.

3.4.1 Neurons in ANN:s

In the artificial neuron model depicted in Figure 7, there are several steps that take place in order to produce an output value. First, the inputs $x_1, x_2, ..., x_n$ are multiplied by their respective weights $w_{1j}, w_{2j}, ..., w_{nj}$. The resulting values are then summed together, along with a bias term b, to produce the weighted sum net_j (depicted as “transfer function” in the figure).

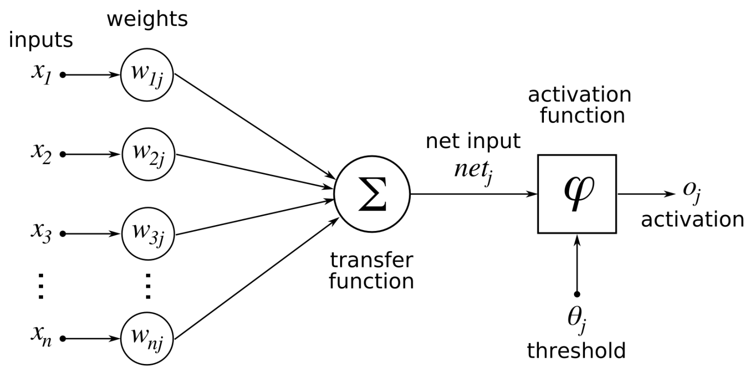


Figure 7: Schematic diagram of an artificial neuron model. Image taken from [17]

This weighted sum is then passed through an activation function, which produces the output value y.
From a scientific perspective, artificial neurons model the behaviour of biological neurons by computing a weighted sum of their inputs, applying an activation function to the result, and then outputting a signal. Mathematically this can be expressed as $y = f(\sum_{i=1}^{n} w_i x_i + b)$, where y is the output, f is the activation function, w_i are the weights, x_i are the inputs, b is the bias term, and $\sum_{i=1}^{n} w_i x_i + b$ is the weighted sum. The choice of activation function can have a significant impact on the performance of the artificial neuron model, and different activation functions may be more suitable for different types of tasks. The choice of activation function will be discussed further under the section for Convolutional Neural Networks.
To find the weights and biases the iterative optimisation algorithm gradient descent is used. The goal of training is to find the optimal weights and biases for the neural network that minimise the loss function. The weights and biases are initialised with random values and updated iteratively using the gradient descent algorithm.
The gradient descent process involves computing the gradient of the loss function with respect to the weights and biases, and then adjusting the values of the weights and biases in the opposite direction of the gradient to minimise the loss function. The update rule for the weight W at each iteration t is given by:

$$W_{t+1} = W_t - \alpha \nabla_W L(W_t) \tag{11}$$

where α is the learning rate, which controls the step size of the weight update, $\nabla_W L(W_t)$ is the gradient of the loss function with respect to the weight, and $L(W_t)$ is the loss function evaluated at the current weight.
The gradient descent process can be slow and may converge to suboptimal solutions. This is often addressed by momentum and adaptive learning rates.

Momentum involves adding a fraction γ of the previous weight update to the current update, which helps to speed up the convergence process and overcome local minima. The update rule for the weight using momentum at iteration t is given by:

$$V_t = \gamma V_{t-1} + (1-\gamma)\nabla_W L(W_t)\ W_{t+1} = W_t - \alpha V_t \tag{12}$$

where V_t is the velocity of the weight update at iteration t.
In this paper we use the Adaptive learning rates, Adam (Adaptive Moment Estimation) to achieve faster convergence and better generalisation. Adam uses estimates of the first and second moments of the gradients to adaptively adjust the learning rate for each weight. The update rule for the weight using Adam at iteration t is given by:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla_W L(W_t) \tag{13}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)(\nabla_W L(W_t))^2 \tag{14}$$

$$\hat{m_t} = \frac{m_t}{1-\beta_1^t} \tag{15}$$

$$\hat{v_t} = \frac{v_t}{1-\beta_2^t} \tag{16}$$

$$W_{t+1} = W_t - \alpha\frac{\hat{m_t}}{\sqrt{\hat{v_t}}+\epsilon} \tag{17}$$

where m_t and v_t are the estimates of the first and second moments of the gradients at iteration t, β_1 and β_2 are the decay rates for the first and second moments, $\hat{m_t}$ and $\hat{v_t}$ are bias-corrected estimates of the moments, and ϵ is a small constant to prevent division by zero.
Training the neural network involves running multiple epochs, where each epoch involves passing the training data through the network, computing the loss function, and updating the weights and biases using the gradient descent algorithm. The weights and biases are updated using the average gradient across a randomly sampled batch of the data.
The most commonly used cost functions for ANN:s include mean squared error (MSE), binary cross-entropy, and categorical cross-entropy, which are used for regression and classification tasks, respectively. The choice of cost function depends on the specific application and type of output. Cross-Entropy is preferred over other cost functions for classification tasks because it penalises high confidence in the incorrect output class, and since this paper focuses on a classification problem cross-entropy will be looked at closer.
The binary cross-entropy cost function is used for binary classification problems. It is given by:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m} y^{(i)}\log(\hat{y}^{(i)}) + (1-y^{(i)})\log(1-\hat{y}^{(i)}) \tag{18}$$

where m is the number of training examples, $y^{(i)}$ is the true output for the i-th example, and $\hat{y}^{(i)}$ is the predicted output for that example.
The categorical cross-entropy cost function is used for multiple-class classification problems. It is given by:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{K} y_j^{(i)} \log(\hat{y}_j^{(i)}) \tag{19}$$

where K is the number of classes, $y_j^{(i)}$ is the true output for the i-th example and j-th class, and $\hat{y}_j^{(i)}$ is the predicted output for that example and class.

3.4.2 Convolutional neural networks

Convolutional Neural Networks (CNN) has emerged as a powerful and widely used deep learning approach for image classification tasks. They are specifically designed to process images and extract meaningful features automatically, making it highly effective for object recognition.
The key motivation behind using CNN for pollen image classification is the ability to capture local patterns in an image. CNNs employ convolutional layers that apply filters to input images, allowing them to learn and extract relevant features hierarchically. These features are then combined through pooling layers to reduce spatial dimensions and further abstract the information. Finally, fully connected layers use these extracted features to make class predictions as in ANN:s.
The theory behind CNNs is based on Goodfellow et al. in the book "Deep Learning" [10], which provides a comprehensive understanding of the principles of CNN. In the following sections, we will delve into the details of CNN architecture, including convolutional layers, pooling layers, and fully connected layers.

3.4.3 Structure

The image provided in Figure 8 shows a schematic diagram of a standard CNN. This neural network is used in image recognition tasks due to its ability to effectively handle spatial data. In the example the first layer is a convolutional layer, which applies a set of filters to the input image to extract features. These filters are learned during the training phase of the network. The output from the convolutional layer is then passed through a non-linear activation function, such as ReLU (rectified linear unit), which introduces non-linearity into the model. This is followed by a pooling layer, which reduces the dimensionality of the feature maps by performing down-sampling. Common type of pooling operation include average pooling and max pooling, with the latter being used and discussed further down in this paper.

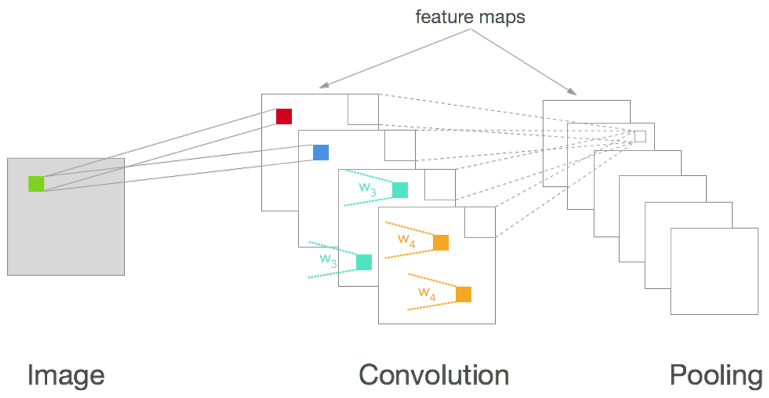


Figure 8: Schematic diagram Convolution neural network. Image taken from [18]

The CNN takes in images, in the form of a 3-dimensional array with the shape of CHW, where C represents the number of colour channels, H represents the height of the image, and W represents the width of the image.
The number of colour channels can vary depending on the type of image. For example, greyscale images have only one colour channel, while RGB colour images have three colour channels (red, green, and blue). Since the images of the Pollen are in RGB we can use all three channels, however since when typically identifying Pollen we would not look at the colour we will also try transforming the images to greyscale and compare the results.
The height and width of the image also vary depending on the resolution of the image. For example, an image with a resolution of 256x256 pixels would have a height and width of 256. If the images that are fed into the network are of different sizes the images need to be resized to a fixed size in order for the CNN to process these images.This ensures that all images have the same dimensions, allowing the network to be trained on a consistent set of inputs.
Once the images are pre-processed and formatted as a CHW array, they can be fed into the CNN for feature extraction and classification. The CNN then applies a set of filters to the input image, learns features and patterns from the image data, and outputs a prediction based on the learned features.
The final layers of the CNN consist of fully connected layers, which map the extracted features to the output classes in the way discussed in the section about ANN.

3.4.4 Choice of Activation functions

In this section we discuss and motivate the choice of activation function for the hidden layers and the output layer of our neural network. For the hidden layers of our network, we have chosen to use the Rectified Linear Unit (ReLU) activation function. ReLU has several advantages over other activation functions, including computational efficiency and the ability to introduce sparsity into the model. This is particularly useful for image classification , where the input data can be highly non-linear and complex. Compared to other activation functions such as sigmoid and tanh, ReLU is computationally more efficient since it involves a simple comparison operation and does not require expensive exponentials or trigonometric functions. ReLU also introduces sparsity to the model, reducing the number of parameters and improve the model's generalisation performance. This is because ReLU sets all negative inputs to 0, effectively removing them from the model's computation. The ReLU function is defined as:

$$f(x) = \max(0, x) \tag{20}$$

where x is the input to the function, and $\max(0, x)$ returns the maximum of 0 and x.
One key advantage of ReLU is that it is a piecewise linear function. When x is positive, $f(x)$ is simply equal to x. When x is negative, $f(x)$ is equal to 0. This allows the model to efficiently focus its attention on the most relevant parts of the input data, while ignoring the less important features.
For the output layer of our network, the Softmax activation function is used. Softmax is commonly used for multi-class classification tasks, where the goal is to assign a label to an input data point from a set of possible labels.
Mathematically, the Softmax function is defined as:

$$f_j(z) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \tag{21}$$

where z is the input to the function, and K is the number of possible labels. The Softmax function produces a probability distribution over the possible labels, with each element of the output vector representing the probability of the input belonging to that label.
Softmax is a useful choice for the output layer because it ensures that the output probabilities sum to 1, making it easier to interpret the output of the network. Additionally, Softmax can handle multi-class classification tasks with a large number of classes, which can be difficult to handle with other activation functions. It does however have a couple of weaknesses one of such is when classifying "new data". For the trained model a fixed number of pollen classes are used, however when later using the trained model to identify and classify pollen we may encounter new pollen species or even objects that are not pollen at all. In these cases, the Softmax output may still assign high probabilities to some of the known classes, even if the sample is not truly a member of any of those classes. This may lead to incorrect predictions and poor performance of the classifier.

3.4.5 Convolution layers

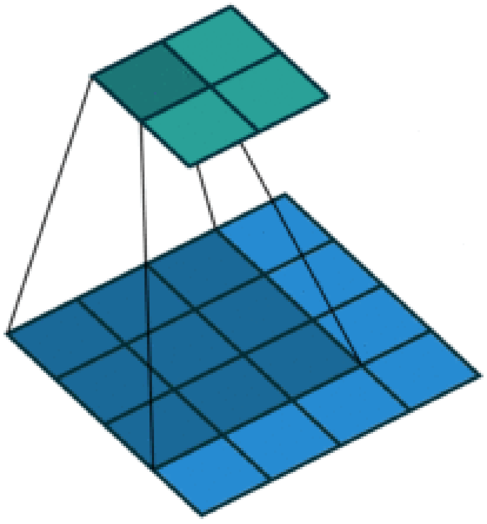


Figure 9: Convolutional operation. Image taken from [19]

Figure [19] depicts the convolution operation with no padding and no strides. It demonstrates the process of applying a 3x3 kernel filter to a 5x5 input image, computing the dot product between the overlapping pixels of the input image and the kernel filter, and generating a 3x3 output image.
The convolutional layer is a fundamental component of the CNN. It is used to extract meaningful features from raw input data by convolving a set of learnable filters over the input and passing these on further into the network. The convolutional operation is defined as follows:
Let I be an input image represented as a $n \times n$ matrix, and let K be a filter or kernel represented as a $k \times k$ matrix. The convolution of I and K is denoted as $I * K$, and is defined as follows:

$$(I * K)_{i,j} = \sum a = 1^k \sum_{b=1}^{k} I_{i+a-1,j+b-1} K_{a,b} \tag{22}$$

In this definition, i and j represent the spatial location of a pixel in the output feature map, and a and b represent the spatial location of a pixel in the input image.
This operation slides the filter over the input image, performing an element-wise multiplication at each location and summing the results to produce a single output value for each spatial location. By repeating this process for each location in the output feature map, the convolutional operation

produces a new image-like representation of the input data that captures relevant spatial features.
Mathematically, the convolution operation can be defined as follows:
Given an input image or signal X with dimensions $H \times W \times C$ (height $\times$ width $\times$ number of channels) and a filter or kernel K with dimensions $K \times K \times C$ (kernel size $\times$ kernel size $\times$ number of input channels), the output Y can be computed as follows:

$$Y(i,j,k) = b(k) + \sum_{p=1}^{K} \sum_{q=1}^{K} \sum_{c=1}^{C} X(p,q,c) \cdot K(i-p+1, j-q+1, c, k) \quad (23)$$

where i and j represent the spatial position in the output feature map, k represents the channel, $b(k)$ represents the bias term for the k-th channel, and p and q are the spatial indices of the filter. The summation is taken over all input channels and filter indices.
In this equation, the bias term $b(k)$ is added to the output of each filter to introduce a degree of freedom in the model. The parameters of the filter are learned through the backpropagation algorithm during the training process.

3.4.6 Max pooling

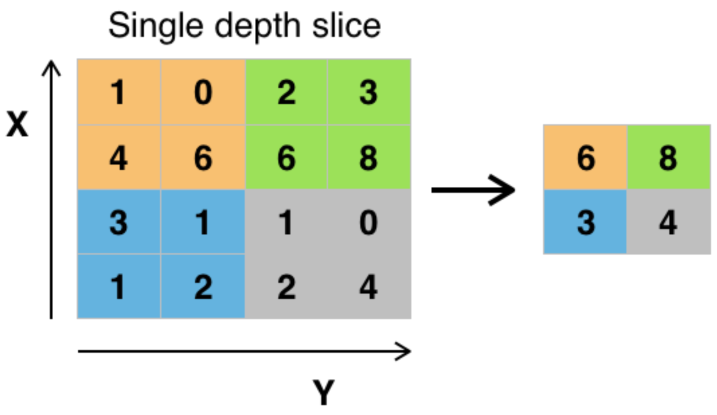


Figure 10: Max Pooling operation. Image taken from [20]

Max pooling is used to reduce the spatial dimensions of the feature maps. The operation works by partitioning the feature map into a set of non-overlapping rectangular regions or pooling windows, and then replacing the values in each window with the maximum value.
In Figure 10, we can see an example of max pooling with a pooling window of size 2x2 and a stride of 2. The pooling window is applied to each non-overlapping 2x2 block of the input feature map, and the maximum value in

each block is selected to produce the output feature map. In this example, the input feature map has a size of 4x4, and the max pooling operation reduces its size to 2x2.
Max pooling is preferred over average pooling in this paper since it preserve the most important features in the input data. This is because the maximum value in each pooling window represents the strongest activation in that region, which often is a good indicator of the presence of relevant features in images. In contrast, average pooling computes the average value in each window, which can dilute the importance of strong activations and reduce the ability of the network to distinguish between important and unimportant features.
Another advantage of max pooling is that it can help to reduce overfitting by introducing a form of spatial invariance. This is because the max pooling operation selects the most salient features in each region, regardless of their precise location within the pooling window. This can help to reduce the sensitivity of the network to small translations or distortions in the input data, which can be a common source of overfitting.

3.4.7 Fully connected layers

Fully connected layers, often called dense layers, are the traditional artificial neural network (ANN) layers that are used in a CNN. The fully connected layers are placed at the end of the network, after the convolutional and pooling layers, to process the features extracted from the input image.
After the last pooling layer a flattening layer is applied, which reshapes the 3D feature maps into a 1D vector. This flattened vector is then passed through the fully connected layers, which are responsible for learning the non-linear relationships between the extracted features and the targeted pollen classes.
The connections in a fully connected layer are dense, meaning each neuron in the previous layer is connected to every neuron in the current layer. Finally, the last fully connected layer in the CNN is a dense output layer with a softmax activation function, which produces the probability distribution over the different classes.

3.4.8 Image augmentation

A technique for enhancing the performance of deep learning models in image recognition tasks is image augmentation. It involves applying various transformations to the original images to generate new training samples, thereby exposing the model to a broader range of variations and improving its ability to generalise to unseen data.
In our implementation, we have utilised several image augmentation techniques, Specifically:

- **Rescaling**: By rescaling the pixel values of the images dividing them by a constant value. This normalisation helps to bring the pixel values into a range between 0 and 1, facilitating the model's learning process without being affected by differences in the original pixel value scales across images.

- **Horizontal flipping**: By mirroring the images horizontally, introducing variations in the orientation of objects in the image. This aids the model in learning to recognise pollen particles from different viewpoints.

- **Vertical flipping**: By flipping the images vertically, introducing variations in the position of pollen particles in the image.

- **Rotation**: Rotating the images, introducing variations in the orientation of pollen particles. This helps the model learn to recognise pollen particles with different rotations.

- **Validation data**: Validation split, for model evaluation during training. This validation data is used to assess the model's performance and make decisions on model selection.

The specific choice of these augmentations is motivated by their potential to improve the classification performance of the deep learning model for the microscope images of the pollen. Rescaling normalises the pixel values, horizontal and vertical flipping introduces variations in object orientation and position, and rotation helps the model learn to recognise pollen particles with different orientations. The use of validation data allows for model evaluation and selection during training, knowing features can be translated to new images.

3.4.9 Model evaluation

In this section about model evaluation we discuss how our CNN models are evaluated using the accuracy and loss metrics, also discussing how to avoid overfitting our model.
Accuracy is a metric for evaluating the performance of a CNN. It is calculated as the percentage of correctly classified instances out of the total instances in the test set. The mathematical formula for accuracy is given by:

$$Accuracy = \frac{\text{Number of correctly classified instances}}{\text{Total number of instances}} \tag{24}$$

Loss quantifies the error in the model's predictions during training. It is computed based on the predicted outputs and the actual outputs using the cross-entropy loss which is suited for multi-class classification,The formula for cross-entropy loss is given by:

$$\text{Cross-Entropy Loss} = -\sum y \log(\hat{y}) \tag{25}$$

where y is a dummy vector representing the true class label, and $\hat{y}$ is the predicted probability distribution over all classes.
Overfitting occurs when a CNN becomes overly complex and memorises the training data, leading to poor generalisation to new, unseen data. One common way to detect overfitting is through the analysis of accuracy and loss curves during training.
During training, the accuracy and loss are computed for both the training set and the validation set at each epoch. The accuracy and loss curves are plotted over the epochs to visualise the performance of the model. If the accuracy of the training set keeps increasing while the accuracy of the validation set starts to degrade or remains stagnant, it may be an indicator of overfitting. Similarly, if the loss of the training set keeps decreasing while the loss of the validation set starts to increase or remains stagnant, it may also indicate overfitting.

4 Methods

This section, will provide a detailed description of the implementation of the tools and techniques mentioned in the Method's Background section. The section provides the methods used of the two main parts of the thesis: segmentation and classification. The focus will lie on explaining the specific steps undertaken in our methodology, the work flow. Additionally information about programming language, the code, packages and selection of different hyper parameters will be included in this Section. Presentation and conclusions made during the work will be left and presented in the following sections.

4.1 Image exploration

The first steps taken was an explorative view of the images before performing segmentation. The numerical RGB values (Red, Green, Blue pixel values) were extracted from the images to analyse our images statistically, giving a deeper understanding of the visual content and properties of the images. By taking the RGB values through a slice on the y-plane of an image we can obtain the colour information for that particular row in the image. This can be useful to analyse how the colour values change for our pollen grains in contrast to other objects that may appear and the general background of the image, this is done in Figure 11. The results show that when the pollen appear on the image the value of the red is significantly higher than for the green and blue pixel values, while other object that are darker and the background have similar pixel values for all three colour channels.

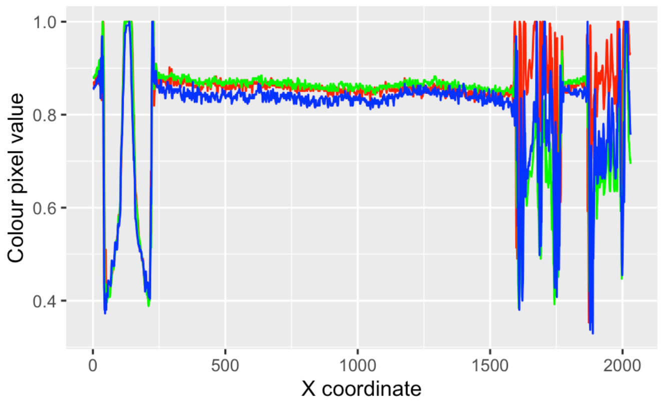


Figure 11: RGB Pixel values from a Slice through y-axis Corylus image in Figure 2. First going through the black circle on the bottom left and then two Corylus pollen on the bottom right of the picture

From Figure 11 it can be seen how the colour values changes when the image pixels are on a pollen grain compared to some foreign dark object. Since the red value is increased and larger than the blue and green colour pixels compared to the black object where the colour values are at a similar level to each other, a potential solution could be to create a grey image by finding the difference between the red pixel values to one of the other 2 colours (which we call “Red difference”).

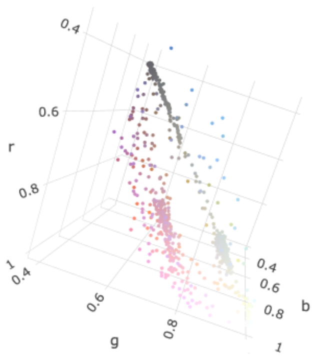


Figure 12: 3D plot of the color values of the Corylus image in 2. The plot shows the RGB values of each pixel in the image, where the X, Y, and Z axes represent the red, green, and blue colour channels, respectively.

The Red difference method is one way to create a grey image using discrimination, another suggestion is drawing a separating plane motivated by Figure 12. The figure indicates that there is a possibility of finding ideal planes, that separate pollen from other background features.

4.2 Segmentation

The segmentation process and all programming was conducted in R [11] along with graphs, functions and tools from tidyverse [12]. Firstly we compare the red difference method, the unsupervised learning method PCA to extract the important features with a separating plane for each image and Supervised learning LDA by drawing masks for pollen. The following steps include thresholding, post-processing and then comparing the watershed algorithm and K-means algorithm before using the favoured method to extract. The imager package in R [13], was used for used for the image processing, thresholding, and Watershed segmentation. Once extracted the smaller images containing only a single pollen particle, are added to a library for each pollen type.

4.2.1 Image processing

The image processing step included pre-processing, thresholding and post-processing. In the pre-processing step want to find a function $y = f(r, g, b)$ that maximises the contrast between pollen and non-pollen.
As mentioned in the method's background the images contain pollen particles on a microscopic slide along with other potential foreign particles. This creates a challenge in the segmentation process. To reduce the noise of other particles, and enhance the quality of our images, we perform pre-processing of the images. In previous work, a common practise when extracting individual pollen from the larger images containing multiple pollen was to convert the colour image to greyscale using method similar to the "greyscale" function in the imager package [13] in R. The function works by taking an RGB image and converting it into a single-channel greyscale image, by calculating the luminance of each pixel in the RGB image using the following formula:

$$\text{luminance} = 0.2126 \times \text{red} + 0.7152 \times \text{green} + 0.0722 \times \text{blue} \tag{26}$$

These are standard coefficients used for converting an RGB image to greyscale, as they take into account the relative luminance of each colour channel. An obstacle with the extraction is that the segmentation images may contain 'dirt' or other objects that are not pollen. The issue with this simple method of converting to greyscale is that it is designed to capture as much of the image as possible, which is not what we are looking for. To highlight the pollen particles in the images, a noise free grey image is desired, with a white background with darker regions appearing where the pollen grains are located.

To find the best pre-processing algorithm an extensive trial with a number of images from each pollen type were compared for the LDA algorithm which was implemented manually in R, PCA using the 'prcomp' function in R and the red difference were tested and compared against one another. After testing, the red difference was selected to create the grey images. To reduce noise and improve image clarity, we applied the 'isoblur' function with a radius of 10. This function implemented an isotropic blur, smoothing the image by averaging the pixel values in the local neighbourhood. By doing so, we achieved a reduction in unwanted noise from differences in colour on the inside of each pollen grain, resulting in cleaner images.
After converting each image to greyscale versions, thresholding was applied to the images to create a binary version. This was done using the 'threshold' function in the imager package. This means we find the threshold value K such that we classify pollen if $f(r, g, b) < K$, and non-pollen if $f(r, g, b) > K$. To address any gaps or holes within the segmented objects, we utilised the 'fill' function with a structuring element size of 15. This function performed a morphological operation called closing, which consisted of dilation followed by erosion. By applying this operation, small gaps or holes were filled in the suspected pollen regions, ensuring a more complete representation of their boundaries.
Furthermore, we employed the 'clean' function with a structuring element size of 3 to eliminate small isolated objects in the image. This function performed a morphological operation known as opening, which involved erosion followed by dilation. By applying this process, we effectively removed small undesired elements, resulting in a more accurate representation of the target pollen grains with less outside noise.

4.2.2 Pollen extraction

With the binary image, the next step is to identify each individual grain and create a bounding box. Thereafter using the location of the bounding box to extract all image slices of the pollen grain and then saving these as JPEG images in the corresponding folder for the extracted pollen type. Since the pollen particles are not necessarily nicely spread out and none overlapping a simple connected components separation of the binary images alone is not sufficient for accurate segmentation since this would lead to a large number of pollen grains in the same bounding box. To address this challenge, exploration of alternative approaches using watershed and k-means were manually tested to extract individual grains.
The 'watershed' function in imager was used to create the watershed mask. To create the watershed an image containing the "sure foreground" was created. This was done by a 'distance_transform' function in R that calculates the distance to the closest dark area (pixel value 0) from each white pixel, pollen particle. This is done to determine areas that definitely are individ-

ual pollen particles. After this step each region from the sure foreground is coloured using the 'label' function performing connected component analysis to find individual grains. The watershed algorithm is then applied to determine the unknown regions between the sure foreground and background to determine which grain the unknown region belongs to, before creating bounding boxes for each individual grain.
After the trial of the watershed algorithm the results were visually compared to the k-means extraction. The k-means algorithm was done using the 'kmeans' function. To determine the appropriate number of clusters for the k-means algorithm in an automated manner, we utilised the clusGap() function available in the 'cluster' package [14]. The clusGap() function calculates the maximum gap statistic, which quantifies the optimal number of clusters based on the within-cluster dispersion. This value was then used in the k argument of the 'kmeans' function. To avoid convergence to local minima and increase the likelihood of finding the optimal solution we set the 'nstart' to a high value of 200 to spread out the starting points of the Algorithm.
Along with a visual comparison a larger study was conducted to determine which of the two methods had the best rate of successfully segmented pollen images. A number of diverse images from different types of pollen grains were selected and the total number of grains were manually counted. Using the two different segmentation techniques, the segmented bounding boxes of the images were then manually inspected and deemed either to be "correctly" or "incorrectly" segmented. The incorrectly segmented images, failed due to not actually containing a grain, containing multiple grains or only containing a partial part of a grain, while the correctly segmented images were images containing one full grain.
After the testing the k-means extraction method was chosen and executed on all the images, the extracted images were visually inspected to ensure higher quality of pollen pictures for the network training and to remove the occasional faulty extracted images. Since the pollen grains often were placed on the edge of the images, many pollen halves were segmented. This lead to a high number of images needing to be manually deleted. After the manual handling of the images a total of 23600 different pollen grains remained.

4.3 Sharpest image

There is a large choice of image depths to feed into the Network, however since some of the depths of the image may be blurry they might not be appropriate or capture the necessary features of the grain. One choice is to feed the compiled images, another option could be to find the sharpest focus depth and use these for training. While the compiled image would capture the overall features in all depths of the image, some of the depths might be out of focus and could distract from the important characteristics of the grains. Which of the depths that is sharpest depends on aspects such as where the pollen lays, in direction and this differs largely. So we need to compare the set of different depths of the microscopic images and find the sharpest versions. To do this we implement an algorithm that uses the concept of image sharpness with image gradients as described by Gonzalez and Woods [8]. The gradient of an image is a measure of the rate of change of the image intensity at each pixel, and the general thought is that the image with the largest differences between pixel values is the sharpest. The sharpness is therefore estimated by the average gradient magnitude. The resulting set of images consists of one image chosen from all depths, for each grain.

4.4 Classification

In this part the process of the build of the neural networks are presented, along with the selected hyper-parameters of our CNN models. The preparation of the models presenting input shapes and batch sizes is included, along with different used augmentation and which images were passed through and trained on.The models were trained and evaluated using the Keras package [15] in R.

4.4.1 Image preparation

The classification task consisted of 6 different models, with 3 different sets of images for training for both a grey and colour version of the images. The reason behind training both grey and colour versions of the model is mainly to compare how much the colour of the pollen grains effect the quality of the network, but also to compare how the grey and colour models perform on the mixed preparations to see if the colour plays a factor can when classifying on images from another microscopic slide. All models were trained with the same set conditions and parameters, the only difference is that the grey and colour models have a different input shape.
The first set of the 3 different set of images were the compiled images, where all slices from the different depths of the image were compiled to one, the second set was with all images with the slice that was deemed the sharpest and the third set was a mixture of the compiled and the 2 sharpest images. This means that the 2 first models contain the same amount of images for training while the third mixture model contains 3 times as many as the other two. To clarify, the mixture model trains on images of the same grain multiple times, and does not treat each image as a 3-dimensional vector for each grain. To avoid the same pollen grain being classified multiple times in the validation split, the validation images from the mixture images were replaced with the validation split for the sharpest images. To avoid the same pollen grain being included in both the training and validation

data, the validation split was made at the bottom of the dataset and not at random.
To enhance the diversity of the training data and improve the model's ability to generalise, data augmentation techniques were applied. The 'image_data_generator' function from the Keras library was utilised for this purpose. The images were rescaled by dividing the pixel values by 255 to normalise them. Additionally, horizontal and vertical flipping, as well as rotation of up to 45 degrees, were applied to the images. A validation split of 0.2 was used to create a separate validation dataset.

4.4.2 Model structure

The first layer in the model was a convolutional layer with 16 filters, a kernel size of 3x3, and a ReLU activation function. The input shape was set to match the target image size of 128x128 pixels with RGB colour channels for the colour image models and a 1 dimensional colour channel for the grey image models. Subsequently, a max pooling layer with a pool size of 2x2 was added to downsample the feature maps. The output was then flattened to be fed into a dense layer with 16 units and ReLU activation. Finally, the output layer consisted of a dense layer with a softmax activation function, which produced the predicted probabilities for the different classes. The model was compiled using the categorical cross-entropy loss function, and the SGD optimiser.
The hyper-parameters were set to the same values for all models to get a fair comparison for the performance of the different input images. The learning rate for the SGD optimiser was set to 0.001, and a decay rate of 1e-6 was applied to reduce the learning rate over time. The momentum parameter was set to 0.9, which accelerated the optimisation process. The chosen hyper-parameters are relatively standard and were used to optimise the model's training process.
The constructed model was trained using the prepared training dataset and validated using the validation dataset. The training process involved multiple epochs, with each epoch consisting of a number of steps defined by the ratio of the training samples to the batch size which was set to 32 images. For consistency 50 epochs were performed for all models to iteratively update the model's parameters and improve its performance. After the training process was completed, the models' performances were assessed by plotting the training and validation accuracy and loss curves and the final validation loss and accuracy. These curves provided insights into the model's learning progress and potential overfitting or underfitting issues.

4.4.3 Assessing Models on mixed preparations

To assess the model performance, a further trial is done by classifying pollen that are found in the same mixture preparation. The preparation contains all types found for the trained models with Rumex the exception. The purpose is to test how well the model performs when the grains are in the same setting, using the same colouring with the same background. Good performance would indicate the model does not rely too much on the preparation's of each pollen type. Segmentation was done using the same process as for the preparations with a single type of pollen. After segmentation, the images were classified manually before using the model to perform classification. The results were then presented in a Confusion matrix. Since manual classification of pollen types is difficult and tedious work, counting was limited to the compiled images. In total there were 227 pollen grain samples.

4.4.4 Highest probability image class

After training the mixed model, another method of classifying each grain of pollen was tested. The thought was to utilise all the different image depths of the pollen grains when classifying instead of just using the sharpest image. However since the different pollen grains have different amounts of image depths the 7 sharpest images and the compiled image of all depths are used. Firstly the softmax probabilities for each of the 8 set of images are determined. Secondly the image depth with the maximum probability are selected for each of the pollen grains giving a probability for each class of each pollen grain. The result are then presented in a table presenting the accuracy, in a plot demonstrating the amount of times each set of images provided the largest probability and the amount for each pollen type.
This method is also done for mixed preparations. Since the mixed preparations contains all depths, the softmax probabilities were instead determined for all depths before finding the maximum probability.

5 Results

The results of this study are presented in two main sections: image segmentation and classification using CNN models. In the image segmentation section, we present the outcomes of the different image processing methods and of the different segmentation algorithms. These results are presented in visual comparisons.
In the classification section, we present the results of our CNN-based classification model. Reporting and comparing the accuracy and loss to assess the performance of the CNN models in classifying the segmented images into the different pollen types.

5.1 Image segmentation

In this section, we present the results of different image processing methods. Beginning by providing visual representations of the PCA, LDA and the “Red difference” grey and binary mask images. This is then followed by the presented masks generated by the K-means Segmentation and Watershed Segmentation.
The results of the image segmentation analysis provide insights into the effectiveness of different image processing methods in segmenting the pollen images and generating masks for further analysis. These findings are discussed in the subsequent sections to draw conclusions and then decide which methods to use to “cut out” individual pollen to use as data for our Classification part of the project.

5.1.1 Image processing

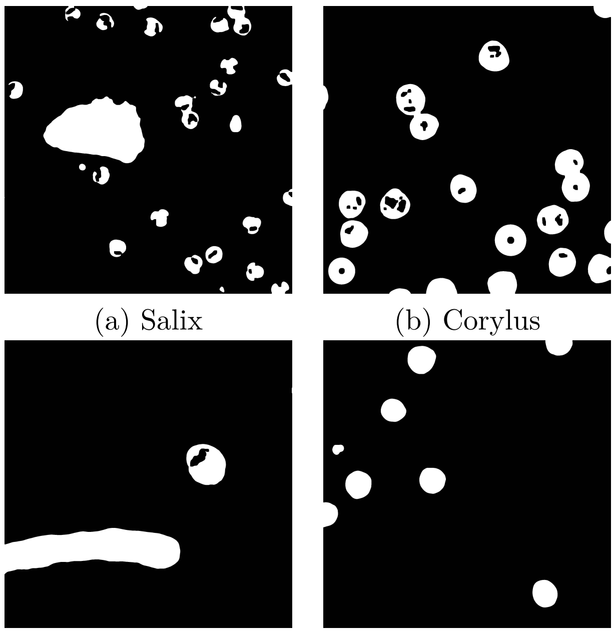

(a) Salix (b) Corylus (c) Quercus (d) Betula

Figure 13: A grid of four Binary images after thresholding grey images from principal component analysis (PCA): (a) Salix, (b) Corylus, (c) Quercus, (d) Betula.

As mentioned the purpose of pre-processing the images is to highlight the pollen grains before thresholding to get the grey image. To determine which of the 3 methods is most successful in this task a large number of images were compared using the different methods. In this section the images provided

earlier in Figure 2 are compared for PCA in Figure 13, for LDA in Figure 14 and lastly for the Red difference in Figure 15.
Since PCA is an unsupervised learning method, each image gets its own hyperplane to determine the important features of the image. This results in Figure 13 grey images that highlight almost all features in the image, including foreign objects.

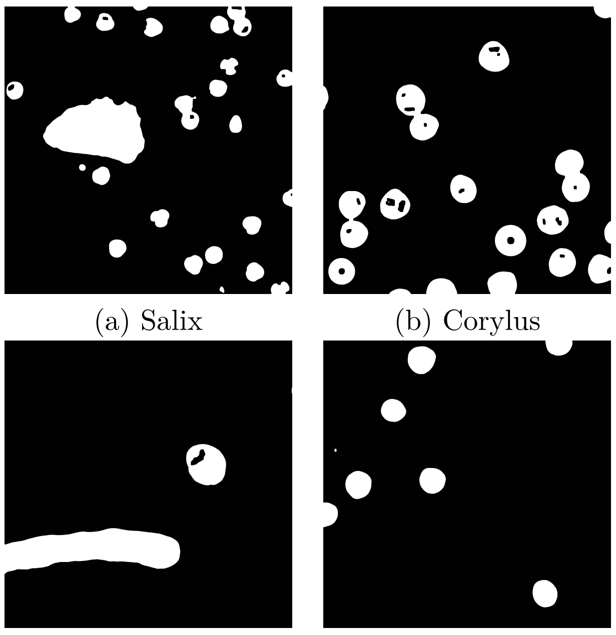
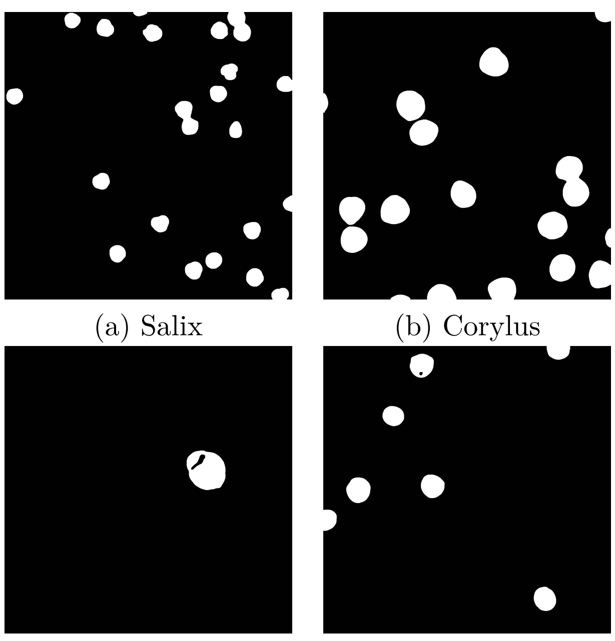

(a) Salix (b) Corylus (c) Quercus (d) Betula

Figure 14: A grid of four Binary images after thresholding grey images from linear discriminant analysis (LDA): (a) Salix, (b) Corylus, (c) Quercus, (d) Betula.

For the LDA in Figure 14 it was determined after drawing masks to discriminate pollen grains from other objects and on a large number of images that the best separating hyperplane which is used for the LDA is determined by:

$$d_i = f(x_i) = -0.16x_{i1} - 0.79x_{i2} - 0.58x_{i3} \tag{27}$$

However as seen in the figure, the foreign objects are not removed. In conclusion neither the PCA or LDA method successfully segregate pollen grains from other objects in the example images. However the overall result from all the sample images used to trial the pre-processing, the LDA did perform better overall.
The clear best performing method for pre-processing was seen in Figure 15. It was largely successful in removing objects that were not pollen grains, as seen in the figures. It successfully removes all objects in the example images and almost all of the objects in the overall trial.

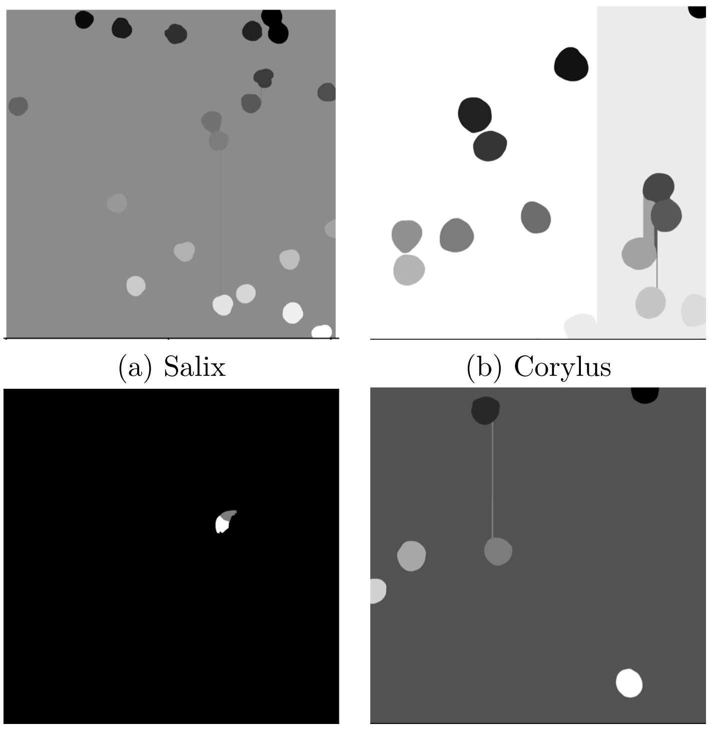

(a) Salix (b) Corylus

(c) Quercus (d) Betula

Figure 15: A grid of four Binary images after thresholding grey images from the Red difference: (a) Salix, (b) Corylus, (c) Quercus, (d) Betula.

5.1.2 Segmentation results

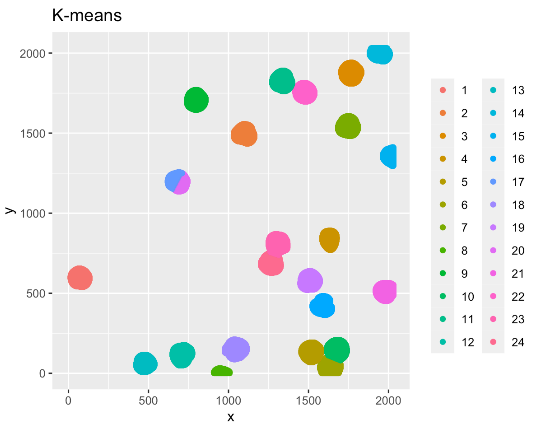
Since the Red Difference was largely successfully in segmenting the pollen grains, it was the selected method chosen for the pre-processing. Moving on the next task was to create bounding boxes for each individual pollen grain, to cut each one out for the classification of pollen type. In some cases it could have been enough to conduct a connected component analysis, which separates and colours all pixels with binary value 1 which are connected. However in many cases and as seen for the examples of Salix, Corylus there are grains that lay next or over each other. Therefore the next comparison will be between to methods that aim to find each individual grain.


(a) Salix (b) Corylus

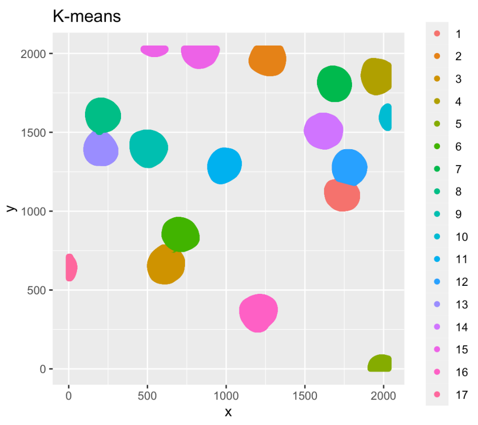
(c) Quercus (d) Betula

Figure 16: A grid of four masks for watershed images: (a) Salix, (b) Corylus, (c) Quercus, (d) Betula.

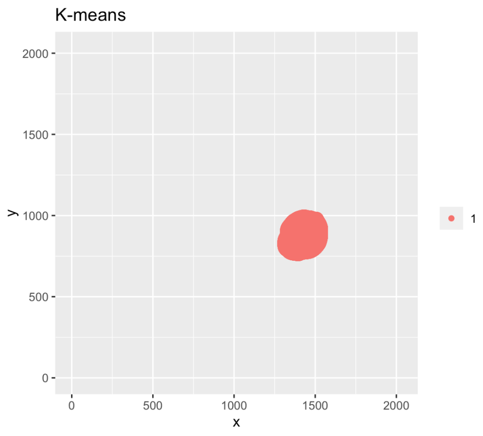
The mask for the watershed algorithm is provided Figure 16. The resulting masks were found to be inconsistent, occasionally working flawlessly and at other instances resulted in colouring all over in places with no pollen. However the method did work well finding individual grains dividing grains next to each other successfully.
The resulting masks in K-means seen in Figure 17 provide good results while occasionally splitting grains. Overall this method was chosen due to largely successfully dividing grains and correctly classifying the number of grains with gap statistic.



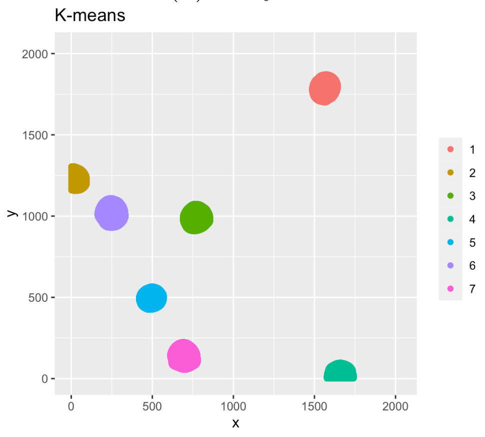
(a) Salix



(b) Corylus



(c) Quercus



(d) Betula

Figure 17: A grid of four masks for k-means clustering: (a) Salix, (b) Corylus, (c) Quercus, (d) Betula.

The results shown in Table 2 compare the performance of the k-means and watershed algorithms for segmenting pollen grains. In total there were 128 pollen grains which could possibly be segmented. Based on the segmentation results, it is evident that the k-means algorithm outperforms the watershed algorithm in terms of correctly segmenting the pollen grains and also detecting the grains. While the k-means algorithm does in fact classify pollen grains incorrectly more often than the watershed algorithm, the overall number of correctly classified pollen easily makes up for this.

Table 2: Segmented Pollen Grains Comparison, giving the total number of pollen available to segment, how many of them that were segmented correctly and incorrectly.

Algorithm	Total Pollen	Correct Segmentation	Incorrect Segmentation
k-means	128	113	13
Watershed	128	95	12

5.2 Classification

In this section, we present the application of our CNN models to the segmented pollen images. The results of our classification performance are presented with a Confusion matrix and the accuracy and loss curves, providing insights into the model's performance during training and validation. The evaluation metrics provide measures of the accuracy and loss of our classification model. Finally the compiled image models are used to classify the pollen grains in the mixed preparations using accuracy.

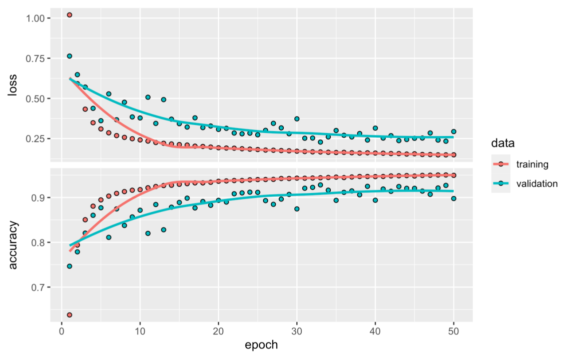
5.2.1 Convolutional neural networks

Firstly the final loss and accuracy values of the 6 different neural networks are provided in Table 3. It can be concluded the models with mixed images have the highest accuracy and lower loss values than both the sharp and compiled images alone. It can also be concluded that the accuracy for the colour image networks is significantly higher than the grey images for all three image types.

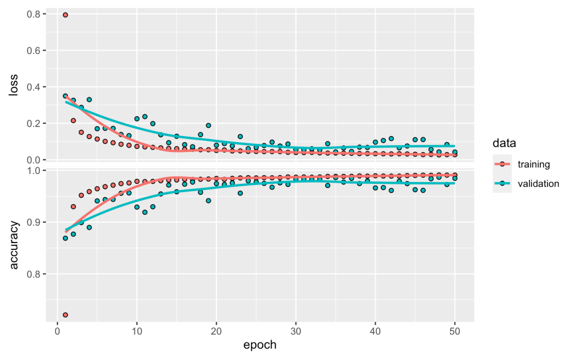
Table 3: Results of CNN models trained on compiled, sharp and mixed images with data augmentation, displayed with validation accuracy and loss.

Training Image Type		**Val. Acc.**	**Val. Loss**
Compiled	**Grey**	0.913	0.334
	Colour	0.958	0.174
Sharp	**Grey**	0.904	0.311
	Colour	0.947	0.139
Mixed	**Grey**	0.925	0.233
	Colour	0.987	0.042

The loss and accuracy curves for the mixed neural network are seen in Figure 18, while the same plots for the sharp and focused (compiled) trained models can be found in the appendix in Figure 20 and Figure 21. The validation loss and accuracy for these follow the training values and don't show any indications of over or underfitting.



(a) Mixed Network Grey)



(b) Mixed Network (Col)

Figure 18: Accuracy & Loss Training Plots for the Mixed Images with sharp validation images CNN

Table 4: Confusion Matrix for the Mixed Colour Model with Sharp validation images

	Alnus	Betula	Corylus	Fraxinus	Quercus	Rumex	Salix	Ulmus
Alnus	86	6	1	0	0	0	1	0
Betula	1	382	0	0	0	0	1	0
Corylus	18	0	529	0	0	0	16	0
Fraxinus	0	0	0	843	0	0	2	5
Quercus	1	2	0	0	456	1	0	0
Rumex	0	0	0	0	0	200	0	0
Salix	4	0	3	0	0	0	1171	0
Ulmus	0	0	0	0	0	0	0	988

The Confusion matrices providing the correct class in the column, against the predicted classes in the rows are provided in Table 4 for the colour mixed model and Table 5 for the grey mixed model. The Confusion matrices for the other 6 models are provided in the appendix.

Table 5: Confusion Matrix for the Mixed Grey Model with Sharp validation images

	Alnus	Betula	Corylus	Fraxinus	Quercus	Rumex	Salix	Ulmus
Alnus	54	19	0	1	0	0	6	0
Betula	14	352	30	0	1	0	3	0
Corylus	4	7	427	0	0	0	17	0
Fraxinus	0	0	5	837	9	0	2	26
Quercus	0	1	0	2	439	0	0	67
Rumex	0	0	0	1	0	201	3	0
Salix	38	11	71	1	0	0	1160	0
Ulmus	0	0	0	1	7	0	0	900

Table 6: Confusion Matrix for the Mixed Model when using 7 sharpest images and compiled images

	Alnus	Betula	Corylus	Fraxinus	Quercus	Rumex	Salix	Ulmus
Alnus	96	1	0	0	0	0	0	0
Betula	0	386	0	0	0	0	0	0
Corylus	13	0	527	0	0	0	0	0
Fraxinus	0	0	0	843	0	0	0	0
Quercus	0	3	0	0	456	0	0	0
Rumex	0	0	0	0	0	201	0	0
Salix	1	0	6	0	0	0	1191	0
Ulmus	0	0	0	0	0	0	0	993

Using the 7 sharpest images and the compiled images gave 0.995 accuracy and the Confusion Matrix can be seen in Table 6. After each set of images set, determine the softmax probabilities for each pollen grain the maximum probability across the sets are chosen for each image. In Figure 19 the number of grains determined by each image set is presented for each type of pollen. In total most Pollen grains were determined by the Focused images, however some pollen types were often classified by the sharpest image or less sharp images.

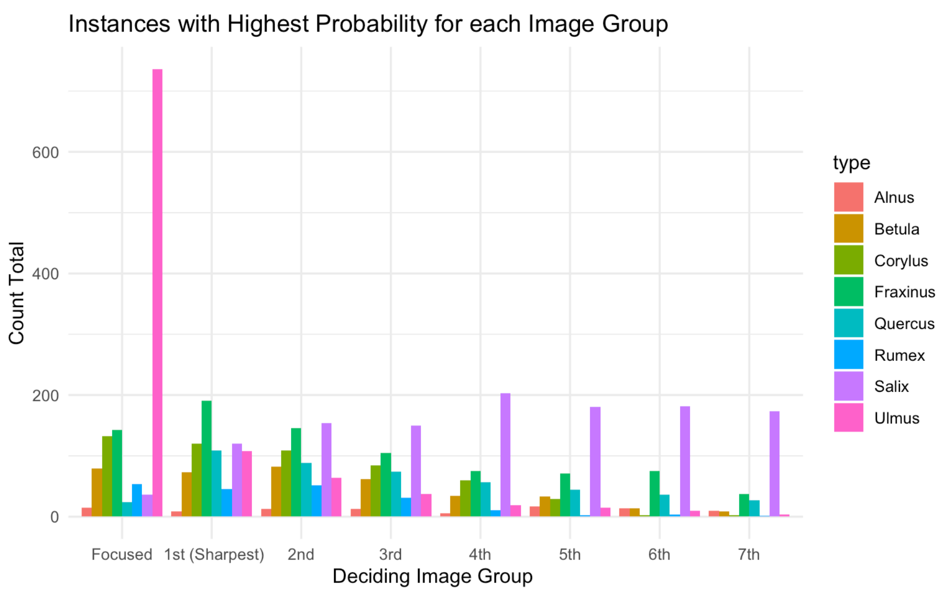


Figure 19: Number of instances where the Highest Probability was found for each set of Images. Colours show the count for each for each pollen type by sets of images.

5.2.2 Mixed preparations result

Table 7: Confusion Matrix for Mixture Preparation for Colour Compiled Model.

	Alnus	Betula	Corylus	Fraxinus	Quercus	Salix	Ulmus
Alnus	30	0	1	1	2	0	2
Betula	0	15	1	0	0	0	0
Corylus	1	0	32	0	2	0	1
Fraxinus	1	0	0	48	1	18	1
Quercus	2	0	0	2	20	1	3
Salix	0	1	0	3	0	25	1
Ulmus	2	1	0	0	2	2	5

Table 8: Accuracy of compiled model for mixture preparation

Model	**Accuracy**
Compiled Grey	0.65
Compiled Colour	0.77
Mixed Colour	0.81

Table 8 shows the accuracy for the models on the mixture preparations. The colour focused shows an accuracy of 0.77 for the colour model with the compiled images and 0.65 for the grey. The Confusion matrix for the colour compiled image model is found in Table 7 and for the grey and colour model in Table 9

Table 9: Confusion Matrix for Mixture Preparation for Grey Compiled Model.

	Alnus	Betula	Corylus	Fraxinus	Quercus	Salix	Ulmus
Alnus	21	4	2	2	2	6	2
Betula	2	7	1	1	0	2	1
Corylus	0	0	31	0	0	1	1
Fraxinus	4	1	0	38	6	6	2
Quercus	7	2	0	5	15	1	2
Salix	0	1	0	7	0	30	0
Ulmus	2	2	0	1	4	0	5

The Mixed model gave accuracy 0.81 and the Confusion matrix is shown in Table 15 in the appendix. When using all image depths the accuracy is increased to 0.85, the Confusion matrix is shown in Table 10.

Table 10: Confusion Matrix for the Mixture Preparation using all image Depths for the Mixed Model.

	Alnus	Betula	Corylus	Fraxinus	Quercus	Salix	Ulmus
Alnus	31	0	0	0	0	1	3
Betula	0	11	0	0	0	0	1
Corylus	0	4	33	0	0	0	0
Fraxinus	0	0	0	52	0	11	1
Quercus	2	1	1	0	26	1	2
Salix	3	1	0	2	0	33	1
Ulmus	0	0	0	0	0	0	5

6 Conclusion

The first step of the image segmentation process, was pre-processing which aimed to extract individual pollen grains from the scanned images using image processing techniques. The effectiveness of different methods was evaluated based on their ability to highlight the pollen grains and remove unwanted objects from the images.
Three different pre-processing algorithms were compared: Principal Component Analysis, Linear Discriminant Analysis and the Red Difference method. The resulting grey images from each method were then thresholded to create binary masks.
The visual comparison of the pre-processed images revealed that PCA and LDA methods were unable to effectively separate the pollen grains from other objects in the images. These methods highlighted almost all features in the images, including foreign objects, which limited their utility in accurate segmentation.
In contrast, the Red Difference method showed promising results. It successfully enhanced the visibility of pollen grains while effectively removing unwanted objects from the images. The binary masks generated from the Red Difference method demonstrated clear boundaries around the pollen grains, indicating successful segmentation. After viewing the image masks it was clear no further comparison on a larger scale was needed, since the Red Difference was highly successful and clearly outperformed the others.
After the creation of a binary mask, segmentation using the watershed and k-means approaches was tested. The watershed algorithm was successful in separating individual pollen grains by identifying the sure foreground and applying connected component analysis. The resulting watershed masks were well segmented, capturing individual pollen grains effectively. It did however fail occasionally and testing on a larger scale of images lead to a few individual pollen grains included in the same image.
The k-means algorithm also captured and segmented individual pollen grains well. Additionally since the k-means algorithm only assigns pixels that have been highlighted in the binary mask as pollen particles, the borders of each assigned pollen grain were well captured in general, with a few grains being split. Overall the K-means algorithm had a higher success-rate in the segmentation process.
Based on these findings, the Red Difference pre-processing method, combined with either the k-means segmentation algorithms, proved to be the most effective approach for accurately extracting individual pollen grains from the scanned images. The resulting segmented images served as high-quality data for the subsequent classification task.
From the results obtained from the CNN models trained on compiled, sharp, and mixed images, several conclusions can be drawn. The inclusion of colour in the image inputs significantly improves the accuracy of the classification

models compared to using greyscale images. This suggests that the artificial colouring clearly plays a role in accurately identifying and classifying different types of pollen grains. Since the colouring of pollen grains may be different between preparations, this is not a desired feature for classification as classification on new preparations or on pollen traps could have bias toward the colouring. The results found for the mixture preparations show significantly lower accuracy, indicating that different preparations play a role in the classification of different pollen types.

Among the different image types, the models trained on mixed images, which consist of a combination of compiled and sharp images, achieved the highest accuracy and the lowest loss values. This indicates that combining different image preparations leads to a more robust and effective classification model. However it does need to be noted that the larger amount of data included in the training of this data most likely is a factor that improves the performance of this model.

Furthermore, the models trained on compiled and sharpest images alone achieved slightly lower accuracy and higher loss values compared to the mixed image types, but still demonstrating high accuracy and relatively low loss values. This suggests that selecting the sharpest image slice from each depth or simply compiling the images provides sufficient information for very accurate pollen grain classification. When studying the Confusion matrices it can be seen that the classification rate for most classes is very high with one exception, the Alnus pollen class. The reasoning behind this is likely not only due to a smaller sample of images than the other classes, but also due to the images being slightly more blurry with less visible features than for the other pollen types.

When classifying over multiple image groups the classification ability of the mixed model increased significantly. When using more image depths more information about the pollen grain can be captured. A concrete example is that the compiled images and the sharpest images may focus on the stronger features like the edges of pollen grains while features such as the texture of pollen not being in focus, when this in some cases can be even more important when classifying pollen types. The result of using more image groups when classifying was encouraging and indicates that even when images are not in best focus they can useful for classification. Most pollen types were heavily classified by the compiled images especially when it came to the Ulmus species, however for the Salix pollen many grains had the maximum probability for some of the less sharp images. An hypothesis of the reason behind this is that one of the strongest features to the Salix pollen is the texture of the grain which is often more visible when the image is less sharp, while the Ulmus pollen has a clear edge which is often highlighted in the sharper or compiled images.

Studying the results for the mixture preparations the grey model did not perform as well as expected, decreasing significantly performance wise compared to the validation images, similarly to the colour model. Since the colours of preparations can vary largely, finding the features on a single colour channel was thought to be able to maintain the classification ability better than a colour model which could potentially overfit based on the colour of the grains. However this was not the case with the colour model still outperforming the grey model. The Ulmus pollen had very low classification in the mixture preparations, indicating the features captured from training were not enough to properly identify Ulmus in a different preparation.

7 Discussion

In this section, we discuss the results obtained in the thesis, compare to previous research and highlight potential areas for improvement. Additionally, we analyse the effectiveness of different methods used.
In terms of segmentation, we trialed a similar approach to Olsson et al. [4] using the watershed algorithm on a binary image. However, a limitation in the segmentation process was identified, as it does not effectively separate pollen grains from other foreign particles in the images. To address this, pre-processing techniques to highlight the pollen grains in the images before thresholding. We compared different pre-processing methods, including the "Red difference" approach, unsupervised learning using PCA, and supervised learning using LDA. The "Red difference" method, which involved calculating the difference between the red pixel values and the other two colors, showed promising results in highlighting the pollen grains.
Regarding the segmentation algorithm itself, we compared the performance of the watershed algorithm with the K-means clustering algorithm. While the watershed algorithm effectively separated connected pollen grains, the K-means algorithm showed a high potential in segmenting individual grains. And since the watershed function in imager did struggle to find the correct edges, the k-means algorithm was preferred.
For the classification task, CNNs were used, which have been widely used in image classification tasks due to their high precision and accuracy. However, acknowledging the challenges associated with creating a suitable dataset for validation, as images of the same grain can vary greatly due to different imaging and staining methods. We compared the use of image augmentation techniques, including horizontal and vertical flipping and rotation, to enhance the diversity of the training data. Additionally, examining the performance of CNNs using both grey images and colour images to assess the impact of colour on classification accuracy. Furthermore, we compared the performance of CNN models trained on different sets of images, including compiled images, images with the sharpest focus depth, and a mixture of both. These comparisons aimed to evaluate the robustness and generalis-

ability of the CNN models in real-world scenarios with pollen classification from the traps.
There are several potential areas for improvement in the field of pollen analysis. Pre-processing and segmentation techniques can be further explored to enhance the accuracy and efficiency of pollen grain detection. One potential idea is to mark individual pixels with the colour specific to pollen grains, rather than considering the entire grain as a region of interest. This approach may help in distinguishing pollen grains from other objects, especially when darker pixels within the pollen introduce difficulties in separating them from surrounding debris.
Exploring alternative clustering methods, such as Gaussian Mixture Models (GMM) or density-based clustering, could offer new insights into pollen grain grouping and identification. Additionally, the adoption of the Hough circle transform could be beneficial for checking the circularity of segmented pollen grains, ensuring they are not cut in half or distorted.
Another potential avenue for research is the usage of a U-Net architecture [6], which would combine the segmentation and classification tasks. By training the neural network with masks, the U-Net structure could accurately locate and segment pollen grains. However, it is crucial to acknowledge the challenge of manually annotating and classifying a significant amount of training data for this approach, also needing the supervision of a pollen classification expert to correctly classify the mixed preparations needed for this solution. Nevertheless, leveraging the power of neural networks could lead to improved pollen grain segmentation and analysis.
With an accuracy of 0.85 the mixed model using all image depths could be used to classify a mixture of pollen grains. However based on the classification results being much more accurate for the validation images and the using the colour images outperforming the greyscale images, future research and work with an automation process of pollen classification would benefit from including images from multiple preparations. This would ensure that even with differences in colour of grains the model would still be able to classify the grain type, avoiding colour being a factor in decision making. There could also be potentially benefit from using different methods of staining.
Furthermore, considering priors based on seasonal variations in pollen characteristics could enhance the accuracy and reliability of the analysis. By incorporating prior knowledge about pollen distributions during different seasons, the classification and segmentation algorithms can be tailored to specific timeframes, leading to more robust results.

References

Related Work & Pollen Articles

[1] Britannica Contributors. Pollen. Encyclopedia Britannica. https://www.britannica.com/science/pollen.

[2] Philipp Viertel and Markus König. Pattern recognition methodologies for pollen grain image classification: a survey. *Machine Vision and Applications*, 33(18), 2022.

[3] Britt Berggren. *Handledning För Pollenanalytiker*. Palmgrens Tryckeri AB, Uppsala, Sweden, 2003. English translation: ”Manual for Pollen Analysts”.

[4] Ola Olsson, Melanie Karlsson, Anna S. Persson, Henrik G. Smith, Vidula Varadarajan, Johanna Yourstone, and Martin Stjernman. Efficient, automated and robust pollen analysis using deep learning. *Methods in Ecology and Evolution*, 12, 2021.

[5] Rafael Redondo, Gloria Bueno, François Chung, Rodrigo Nava, J. Víctor Marcos, Gabriel Cristóbal, Tomás Rodríguez, Amelia Gonzalez-Porto, Cristina Pardo, Oscar Déniz, and B. Escalante-Ramírez. Pollen segmentation and feature evaluation for automatic classification in bright-field microscopy. *Computers and Electronics in Agriculture*, 110:56–69, 2015.

[6] Victor Sevillano, Katherine Holt, and Jose L. Aznarte. Precise automatic classification of 46 different pollen types with convolutional neural networks. *PLoS ONE*, 15(6):e0229751, 2020.

[7] Mihai Boldeanu, Mónica González-Alonso, Horia Cucu, Corneliu Burileanu, Jose Maria Maya-Manzano, and Jeroen Titus Maria Buters. Automatic pollen classification and segmentation using u-nets and synthetic data. *IEEE Access*, 10:73675–73684, 2022.

Books

[8] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Science & Business Media, 2009.

[9] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Prentice Hall, Upper Saddle River, NJ, 2008.

[10] Robert Tibshirani, Guenther Walther, and Trevor Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423, 2001.

[11] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. [Online; accessed 27-March-2023].

R Packages

[12] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2020.

[13] Hadley Wickham, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D'Agostino McGowan, Romain François, Garrett Grolemund, Alex Hayes, Lionel Henry, Jim Hester, Max Kuhn, Thomas Lin Pedersen, Evan Miller, Stephan Milton Bache, Kirill Müller, Jeroen Ooms, David Robinson, Dana Paige Seidel, Vitalie Spinu, Kohske Takahashi, Davis Vaughan, Claus Wilke, Kara Woo, and Hiroaki Yutani. Welcome to the tidyverse. *Journal of Open Source Software*, 4(43):1686, 2019.

[14] Simon Barthelme and David Tschumperlé. *imager: Image Processing Library Based on 'CImg'*, 2021. R package version 1.4.4.

[15] Martin Maechler, Peter Rousseeuw, Anja Struyf, Mia Hubert, and Kurt Hornik. *cluster: "Finding Groups in Data": Cluster Analysis Extended Rousseeuw et al.*, 2019. R package version 2.1.2.

[16] J.J. Allaire, F. Chollet, S. Van Der Walt, T. Atkins, K. Ushey, Y. Tang, M. Kuhn, C.E. McCulloch, D. Vaughan, M. Schwager, et al. *keras: R Interface to 'Keras'*. RStudio, 2021. R package version 2.6.0.

Images

[17] Maurice44. Colored neural network. https://upload.wikimedia.org/wikipedia/commons/4/46/Colored_neural_network.svg, 2019. [Online; accessed 27-March-2023].

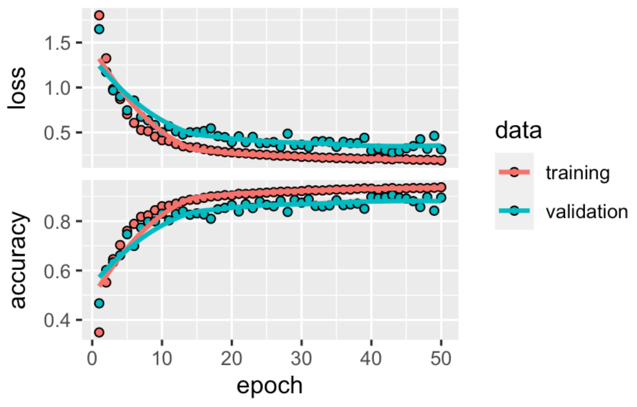
[18] Looxid Labs. Artificial neuron model. https://upload.wikimedia.org/wikipedia/commons/6/60/ArtificialNeuronModel_english.png, 2021. [Online; accessed 27-March-2023].

[19] Adrian Rosales. Convolutional network, 2014. Accessed: April 12, 2023.

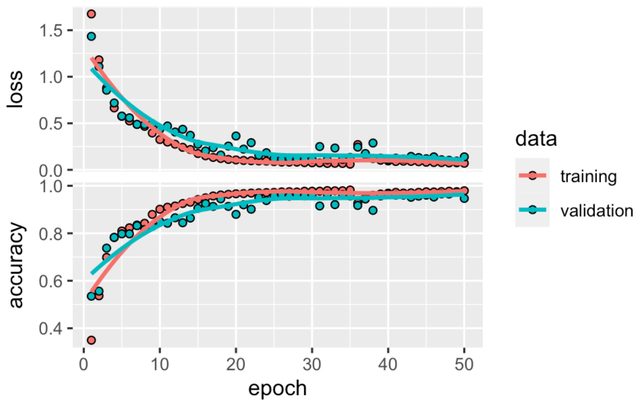
[20] Vincent Dumoulin. Convolution arithmetic - no padding no strides, 2016. Accessed: April 12, 2023.

[21] Adrian Rosales. Max pooling, 2014. Accessed: April 12, 2023.

8 Appendix

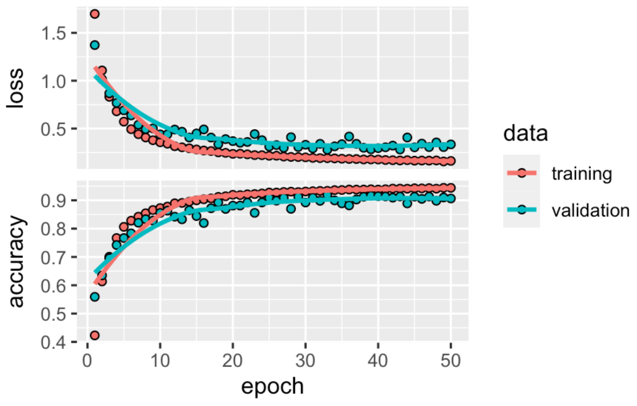


(a) Sharp Network (Grey)

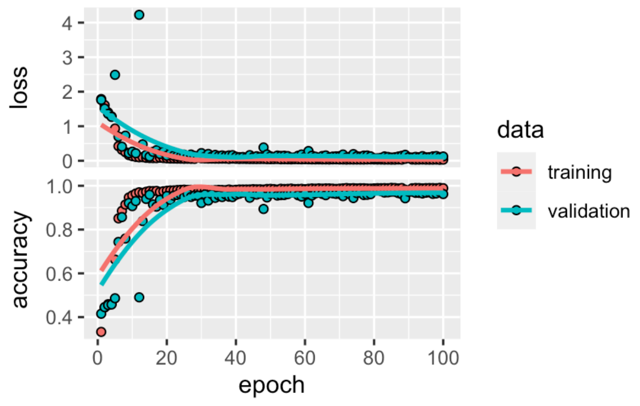


(b) Sharp Network (Col)

Figure 20: Accuracy & Loss Training Plots for the Sharp Images CNN



(a) Focused Network (Grey)



(b) Focused Network (Col)

Figure 21: Accuracy & Loss Training Plots for the Focused Images CNN

Table 11: Confusion Matrix for the Sharp Colour Model

	Alnus	Betula	Corylus	Fraxinus	Quercus	Rumex	Salix	Ulmus
Alnus	83	1	8	0	0	0	0	0
Betula	5	388	3	0	0	0	0	0
Corylus	22	1	518	0	0	0	4	0
Fraxinus	0	0	0	840	0	0	1	2
Quercus	0	0	0	0	456	0	0	0
Rumex	0	0	0	0	0	158	6	1
Salix	0	0	4	0	0	0	1051	0
Ulmus	0	0	0	0	1	0	0	977

Table 12: Confusion Matrix for the Sharp Grey Model

	Alnus	Betula	Corylus	Fraxinus	Quercus	Rumex	Salix	Ulmus
Alnus	26	14	0	1	0	0	49	0
Betula	34	354	29	0	1	0	11	0
Corylus	1	9	428	2	0	0	68	1
Fraxinus	0	1	5	792	0	0	2	11
Quercus	0	1	0	28	425	0	0	29
Rumex	0	0	2	6	0	200	0	0
Salix	26	4	31	0	0	3	1132	1
Ulmus	0	0	1	11	13	0	0	934

Table 13: Confusion Matrix for the Compiled Colour Model

	Alnus	Betula	Corylus	Fraxinus	Quercus	Rumex	Salix	Ulmus
Alnus	48	4	2	0	0	0	54	0
Betula	1	383	0	0	0	0	0	1
Corylus	7	1	492	0	0	0	34	5
Fraxinus	0	0	0	837	0	0	0	6
Quercus	0	1	0	0	454	0	0	2
Rumex	0	0	0	0	0	180	6	0
Salix	54	0	34	0	0	0	1184	0
Ulmus	0	1	5	6	2	0	0	942

Table 14: Confusion Matrix for the Compiled Grey Model

	Alnus	Betula	Corylus	Fraxinus	Quercus	Rumex	Salix	Ulmus
Alnus	51	16	0	1	0	0	46	0
Betula	8	358	41	1	1	0	7	0
Corylus	5	8	451	4	0	0	39	1
Fraxinus	0	0	1	778	0	2	0	6
Quercus	0	1	0	50	428	0	0	28
Rumex	0	0	0	3	0	186	10	3
Salix	46	7	39	0	0	0	1143	0
Ulmus	0	0	1	6	75	0	0	916

Table 15: Confusion Matrix for the Mixture Preparation using the Mixed Model.

	Alnus	Betula	Corylus	Fraxinus	Quercus	Salix	Ulmus
Alnus	27	0	0	0	0	0	4
Betula	1	15	3	0	0	0	1
Corylus	0	0	29	0	0	0	0
Fraxinus	0	0	0	50	0	14	0
Quercus	8	1	2	0	25	1	2
Salix	0	1	0	4	1	30	0
Ulmus	0	0	0	0	0	1	6