Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Fredrik Gyllenhammar

Automated Pollen-Grain Counting

Master's thesis in Computer Science
Supervisor: Professor Keith Downing
June 2021

NTNU
Norwegian University of
Science and Technology

Fredrik Gyllenhammar

Automated Pollen-Grain Counting

Master's thesis in Computer Science
Supervisor: Professor Keith Downing
June 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Abstract

This thesis explores how a CNN based object detection model may be used to localize and classify pollen grains using microscopic imaging data. Pollen counting is a central method in many diverse fields, e.g., criminology, archaeology, and geology. This is a laborious and very time-consuming task that currently requires expert knowledge. From the literature, open questions remain with regards to the complexity needed to solve this problem versus more common object detection tasks. The effects of sharpness within training examples are also unclear. Experiments using a Single Shot Multibox Detection model reveal that the problem is solvable with a fully convolutional model. The regular shape of pollen grains allows for certain simplifications to the model, but the similarities across classes cause a loss in accuracy in smaller model configurations. Excluding un-sharp data from the model's training data causes the model to fixate on sharpness, reducing the model's ability to identify grains that appear less sharp. Training with un-sharp examples seems to allow for a more robust generalization over the features encoded in multifocal data.

Sammendrag

Denne oppgaven utforsker hvodan CNN baserte objektdeteksjonsmodeler kan bruker til å lokalisere of klassifisere pollenkorn ved hjelp av mikroskopisk bilde data. Telling av pollen er en sentral metode innen mange forskellige felt, f.eks. krimimalogi, arkeologi, og geologi. Dette er en møysommelig og veldig tidkrevende oppgave som per nå krever ekspertkunnskap. Fra litteraturen finnes det åpne spørsmål med hensyn til kompleksiteten som trengs for å løse dette problemet i forhold til mer vanlige objektdeteksjonsoppgaver. Effekten skarpheten til treningsekemplene her på modellen er også uklar. Eksperimenter med en 'Single Shot Multibox' deteksjonsmodell viser at problemet er løselig med en fullt konvolusjonell modell. Den regulære formen til pollenkorn tillater visse forenklinger av modellen, men likhetene på tvers av klassene fører til tap av nøyaktighet i mindre modellkonfigurasjoner. Ekskludering av uskarpe data fra modellopplæringen får modellen til å fiksere på skarphet, noe som reduserer modellens evne til å identifisere korn som er mindre skarpe en trenings eksemplene. Trening med uskarpe eksempeler ser ut til å tillate en mer robust generalisering over de ukile attributtene i multifokale data.

Preface

This master's thesis is written for the Department of Computer and Information Science at the Norwegian University of Science and Technology. This project is a continuation of an unpublished specialization project thesis of the same name conducted in the Fall of 2020, which functioned as the literature review for this thesis. That project produced draft versions of Chapters 1, 2, and 3.

I want to thank my supervisor Professor Keith Downing for his invaluable guidance and insightful feedback. I also want to thank Trond Einar Brobakk and The Norwegian Asthma and Allergy Association for their help and expertise with the data collection.

Lastly, I would like to thank the Open Source Software community, without which this project could not exist.

Fredrik Gyllenhammar

Trondheim, June 10, 2021

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Palynology is the scientific study of palynomorphs, a general term for organic-walled microscopic plant and animal remains (Askin and Jacobson, 2003). Because of the resilience of these types of organisms and microfossils, the types of particles studied cover the entire geological timeline, from the earliest organisms of the Proterozoic Era to the allergy-educing grass pollens of today.

Possible applications and use cases are equally as diverse. In criminology, soil samples can place a suspect at a specific location, depending on the composition of pollen and spores found. Geologists analyze rock layers and use the presence and disappearance of palynomorphs to place formations in time. Glacial Ice core samples are analyzed for organic remains to estimate temperature and rainfall over the past 10,000 years. Finally, and most relevant to this thesis, is counting the amount of airborne pollen to forecast conditions for people with allergies.

Methods used in palynology vary, but most seek to identify the composition of palynomorphs in samples taken from nature, be it from glacial ice cores, peat sections from bogs, rock samples from stratigraphic drilling, or collected airborne pollen grains. Common for all these methods is the need for human experts to count and classify palynomorphs with a microscope manually. A single slide can take hours to analyze, directly affecting the potential amount of data collected and analyzed.

From a machine learning perspective, the above describes an object detection task: the general task of locating certain objects within an image and classifying each object. State of the art within this problem space uses Convolutional Neural Networks (CNN), a type of neural network especially suited for image analysis of unprocessed image data. However, research into using this technique to solve counting pollen is sparse, and as of writing, only one partial example exists in literature. Microscopy is a relatively unexplored domain for machine learning and offers unique challenges in need of research. This thesis will explore the varying methods that have been proposed to automatically count pollen grains, as well as other domains where modern machine learning methods have been used on similar tasks.

1.1 Goals and Research Questions

Many of the major advancements within object detection have happened so recently that many possible use cases, pollen counting being one, have yet to be explored. The goal of this thesis can therefore broadly be stated as follows,

Goal *To explore the use of Convolutional Neural Networks in automated pollen counting.*

The primary objective is to build a system that can count pollen grains with an accuracy comparable to that of a human expert. Moreover than just developing a working system, the project aims to establish whether the modifications that have been successfully made to detection systems in related domains also may improve a pollen detection system. This is formalized as the following research question,

RQ1 *Can the computational complexity of a Single Stage MultiBox object detection model be reduced without a loss in precision and recall?*

Computational complexity here refers to the number of trainable parameters that a given model comprises. The Single Stage MultiBox (SSD) model is presented in Section 3.1.2. It is designed as a general object detector capable of detecting any number of objects in an image. **RQ1** postulates that the task of pollen detection will require a less generalized model and that this can be realized by simplifying or removing parts of the model's architecture.

Recent research in pollen classification has questioned how multifocal data may be used to improve the accuracy of models operating on microscopic data, i.e., images from different focal planes instead of single images. This project will explore what impact the sharpness of training data has on an object detection model. This is formalized in the second research question,

RQ2 *Can the accuracy and recall of the model be improved by using multifocal data?*

1.2 Problem Description

As mentioned, one of the primary activities within palynology is counting pollen grains. When magnified, only a section of a slide is visible through a microscope. A sliding window approach must be used to scan the entire surface area of the sample. The data that is collected varies between different applications. With airborne pollen, the slide has been prepared such that the location of pollen grains represents the time interval at which it was collected. In this case, the general location and taxa are recorded, such that the changes in density throughout the day are known.

In the context of machine learning, this can be described as an *image recognition* task. Image recognition is the general task of deciding if an image contains an object of interest, where it is located within the image, and to which class the object belongs. When the main task is to locate one or multiple objects, the task is often referred to as *object detection*, which is a joint regression and classification

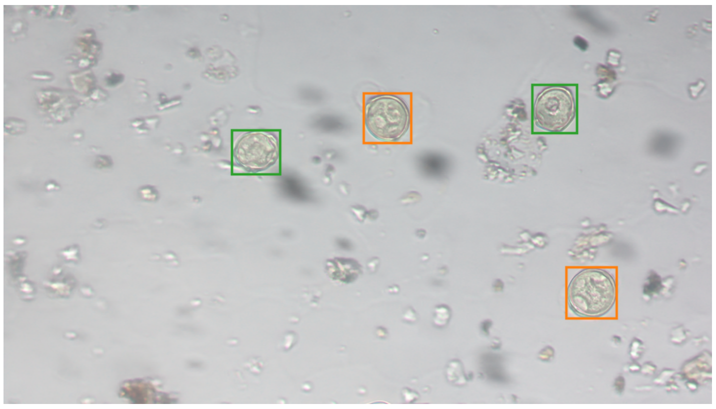


Figure 1.1: An LM image of four pollen grains with ground truth bounding boxes. The image contains two species, corylus and alnus.

problem. The location and dimensions of a rectangle, known as a *bounding box*, which encloses an object, is regressed and its class is identified. Figure 1.1 shows a correct solution to this problem.

1.3 Thesis Structure

The remainder of this document is structured as follows, Chapter 2 covers background knowledge relating to pollen imaging techniques, the composition and functioning of Convolutional Neural Networks, and metrics for measuring performance in object detection tasks. A literature review covering the usage of convolutional networks in object detection, as well as the various methods proposed to classify and detect pollen grains, follows in Chapter 3. Chapter 4 describes the implementation and development of a CNN based object detection system, a sharpness measurement procedure, and the experimental design used to analyze the model. The results of these experiments are provided and discussed in Chapter 5.

Chapter 2

Background

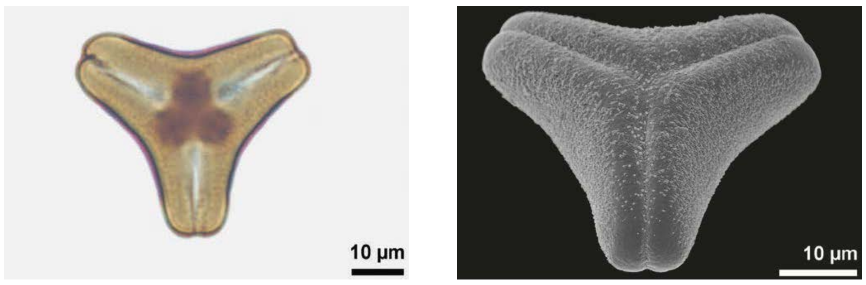
This chapter covers the main theoretical concepts underlying the problem domain and proposed solution. Section 2.1 gives a short overview of the current methods that are in use for pollen counting and the data that is available. Based on this, Section 1.2 formalizes the task as a machine learning problem. A theoretical overview of the main building blocks of modern convolutional neural networks follows in Section 2.2, together with an overview of the metrics used to measure the performance of object detection models in Section 2.4. A basic understanding of the operation and components of a standard feedforward fully connected neural network is assumed for this section.

2.1 Pollen Imaging

There are two main methods of pollen analysis, image-based and non-image-based. Non-image-based techniques employ a host of alternative sensing methods and will not be discussed further in this thesis. Within the image-based methods there are two main imaging techniques.

Light microscopy (LM) describes the method of observing a prepared sample with an optical microscope using visible light. The sample is fixed to a translucent slide and is illuminated with a backlight. It can either be observed through an eyepiece or photographed with an image sensor. An example of a pollen grain is shown in Figure 2.1. Because the grain is semi-translucent, differences in the surface texture can be observed, but only some areas are in focus. A consequence of the high magnification is that the plane of focus is so narrow that only parts of the grain are in focus.

Scanning electron microscopy (SEM) is a very different approach where a focused beam of electrons is used to record the surface topology of a sample. It captures very detailed features of the pollen grain surface but cannot reveal any of the substructures. Because SEM imaging does not depend on focusing light, all parts of the image appear in focus, and the resolution is much higher than what LM can achieve. However, SEM imaging is a more laborious process and requires more



(a) Light Microscopy **(b)** Scanning Electron Microscopy

Figure 2.1: *Aetanthus coriaceus*. Imaged with LM (a) and SEM (b). Halbritter et al. (2018:p. 98) / cropped and rearranged, licensed under CC BY 4.0 URL: `https://creativecommons.org/licenses/by/4.0/`

preparation of the sample. SEM is also not suited for large samples where pollen grains must be observed over the entire slide. This is why LM imaging is the only viable option when the task is to count pollen grains on a slide.

The current standard method for pollen counting is a *sliding window search*. A human expert views a prepared slide through a microscope and systematically searches for pollen grains within the slide. The slide is often partitioned such that the size of the searched area is known; this is then used to estimate the concentration of pollen. A machine learning system should integrate easily into this existing search-based workflow, but it cannot be assumed to be the most effective way to perform the overall goal.

2.2 Convolutional Neural Networks

Convolutional Neural Networks have been in active development for three decades, and the umbrella of what the term encompasses continues to grow. The basic concepts and building blocks have remained relatively unchanged since they were first used to predict handwritten digits in LeCun, B. Boser, et al. (1990). An overview of the basic building blocks will first be given before expanding on each building block. This section will also cover some of the newer concepts that have become commonplace additions to the basic model in later years.

A convolutional neural network consists of stacked and layered operations. There are two types of layers, *convolutional*, and *spatial pooling*. The convolutional layers extract *feature maps* by applying several trainable filters to the input before applying a nonlinear activation function to the result. The spatial pooling layers operate similarly by applying an operation to a receptive field moved over the input feature map. The operation downsamples the input, reducing its spatial dimensions. Figure 2.2 shows an example of a simple CNN model; the convolutional layers control the depth of the activations while the pooling layers control their spatial dimensions. The two layers are stacked alternatingly, with the idea that the complexity of the features extracted increases with the depth of the network.

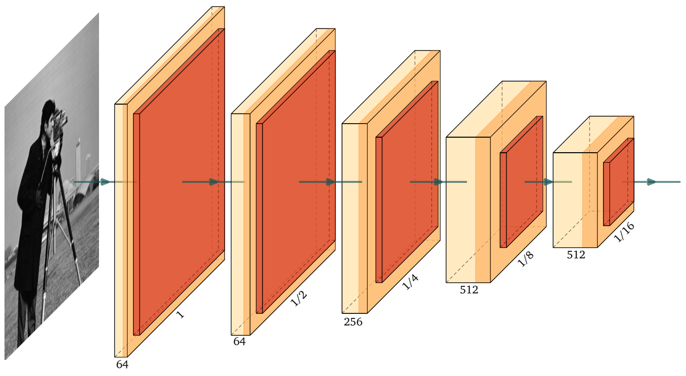


Figure 2.2: The basic architecture of a CNN. Convolutions create feature maps (yellow) which are followed by a non-linear activation (orange). The pooling layers (red) reduce the size of the feature maps. Here, the pooling layers halve the spatial dimensions of the feature maps. Visualization library (Iqbal, 2018).

2.2.1 Convolution

The central concept of the convolutional layer is the *convolution operation*. Let the kernel (w) be a $k \times k$ matrix. This kernel will operate on the output of the preceding layer, x. The output from the convolution can be calculated as follows,

$$w * x_{ij} = \sum_m \sum_n w_{mn} x_{i-m,j-n}$$

Where (m,n) spans the index set of the kernel, which is center originated, i.e., $w_{0,0}$ is the centroid of the kernel. The patch of x involved in the sum at each step is referred to as the *receptive field*. As the operation is repeated for every index of x, the receptive field slides across the input. The resulting output of the convolution is referred to as a *feature map*.

Usually, the input to a convolutional layer contains multiple channels, e.g., an RGB image with three channels representing the red, green, and blue color channels. A stack of kernels is therefore used, one for each input channel. Each channel is convolved with its kernel, and the result is added together across the channels, which produces a single feature map. An example of such a convolution operation is shown in Figure 2.3. This stack of kernels is referred to as a *filter*. For a convolutional layer to produce N feature maps, N filters are needed. It is common to increase the number of filters as the image is continually downsampled through the layers on the neural network.

At the edges of x the sum is undefined because the receptive field moves beyond the bounds of x, causing a reduction in the size of the output. This can be mitigated by *padding* the input. When the receptive field moves beyond the bounds of x, a stand-in value is used instead. This can be visualized as *padding* the input with said value. Zero is often used as the padding value.

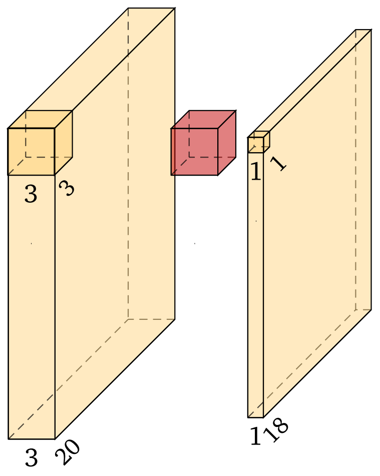


Figure 2.3: Visualization of the convolution operation. In red is a filter containing three 3×3 kernels. The element wise multiplication between the filter and receptive field and subsequent summation produces a single scaler in the feature map. The operation is repeated over the index set of the input, producing the complete feature map.

Dimensionality reduction is also possible using the concept of *stride*, which refers to how the receptive field moves across the input relative to the index of the feature map. In the base case, the receptive field moves by one step for every element in the feature map. With an increased stride, the receptive field 'jumps over' positions for every step in the feature map, thus shrinking its size.

One of the more essential aspects of convolutions arises from the fact that the kernel is applied in the same way over the whole image. This parameter sharing means that features are extracted from the input, regardless of their location (LeCun, 1989). It also reduces the computational complexity involved in training the model.

Because convolution is a linear operation, non-linearity must be added if the network is to be able to approximate a nonlinear function. As with regular fully connected networks, this is achieved by applying an activation function to the feature map. The same activation functions that are commonly used in fully connected networks are also used in CNNs. Because of the depth of the models' architectures in use today, the *rectified linear unit* (ReLU), and its variations, are commonly used.

2.2.2 Spatial pooling

Even though the convolution operation extracts features wherever they exist within an image, a new problem arises when layers are stacked to extract higher-level features from the combination of features below. Local variations in the relative placement of features will significantly impact later filters' ability to combine them. This would have to be accounted for by dramatically increasing the number of filters. LeCun, Bottou, et al. (1998) presents a simple solution to this problem with a *sub-sampling* layer, referred to as a *pooling* layer today, which reduces the dimensions of the feature map by applying a local pooling function, similarly to the convolution operation. Standard pooling functions are maximum and average. The pooling operation is applied to each channel separately, so only the width and height are downsampled. Pooling retains the relative placement of features within the image while allowing the network to ignore smaller variations in the relative configuration of features across all the channels of the feature map.

2.2.3 Cross channel pooling

As mentioned, it is customary to increase the depth of the feature maps as they get downsampled throughout the network. This is necessary if the model is to learn more complex features that may require many layers to be represented in full. Combining information from multiple channels could help build more rich feature maps. This is the proposal in Lin, Chen, and Yan (2014). To enhance model discriminability, they propose a 'Network in Network', a fully connected layer working across the channels. This effectively creates connections between local features across the channels of the feature maps, as shown in Figure 2.4.

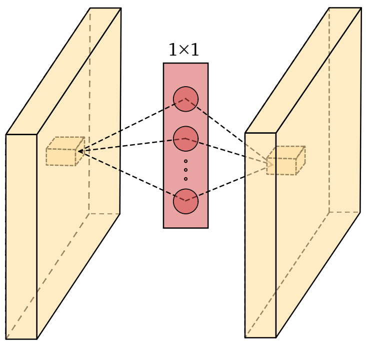


Figure 2.4: Visualization of a cross channel pooling architecture.This simple example models a single fully connected layer using a 1×1 convolution.

The technique is however most commonly used as an optimization that removes computational bottlenecks in deeper networks (Szegedy et al., 2014). By placing a 1×1 convolution with reduced output depth in front of a larger 3×3 or 5×5 convolution, the computational cost is reduced, which allows for much deeper networks. For instance, given an input depth of 500, a $3 \times 3 \times 500$ convolution requires 2,250,000 parameters, but if a $1 \times 1 \times 250$ is used first, the total number of parameters in both layers is only 1,250,000. The technique is now commonplace and featured in all deep CNN architectures.

2.2.4 Batch normalization

As the network trains, the parameters in each layer change. This causes the distribution of each layer's output to shift. As the distribution from previous layers changes, this shift is propagated through to the layers downstream, and so each layer must deal with ever-changing input distributions. To overcome this, lower learning rates and careful parameter initialization is required.

A much more effective solution has been proposed by Ioffe and Szegedy (2015) called *Batch Normalization* (BN). The proposed solution for convolutional networks is to normalize the layers from each convolution independently. Given a layer activation with d feature maps $\mathbf{a} = \left(a^{(1)}, \ldots, a^{(d)}\right)$, each feature map (k) is normalized (pre activation) as follows,

$$\widehat{a}^{(k)} = \frac{a^{(k)} - \mathrm{E}\left[a^{(k)}\right]}{\sqrt{\mathrm{Var}\left[a^{(k)}\right]}}$$

where $\mathrm{Var}[\cdot]$ and $\mathrm{E}[\cdot]$ are respectively the batch variance and batch mean over both the mini-batch and spacial locations of the feature map, thus maintaining the convolutional property of spatial invariance within feature maps. However, this normalization could be undesirable in certain circumstances. For instance, if the inputs of a ReLU activation are normalized, roughly half of the features will be truncated at 0. To account for this, the normalized values are scaled and shifted before activation. Two parameters, $\gamma^{(k)}$ and $\beta^{(k)}$ are introduced for each feature map, and the normalized values are scaled and shifted as follows,

$$y^{(k)} = \gamma^{(k)} \widehat{a}^{(k)} + \beta^{(k)}$$

The parameters are learned together with the original filters and restore the representative power of the layers. By allowing the filters to only focus on learning features, instead of adapting to constantly shifting input distributions, training is accelerated, allowing for higher learning rates.

2.2.5 Data augmentation

Deep learning in general, and deep convolutional networks in particular, require a large amount of data to generalize to a solution properly. Data augmentation is a technique whereby the size of a dataset is artificially increased by applying transformations to the existing data. This has been an important regularization technique and a critical component of many established models, such as ResNet (He et al., 2015) and AlexNet (Krizhevsky, Sutskever, and Hinton, 2017). It is argued by Hernández-García and König that data augmentation alone is more beneficial to training than using explicit regularization such as dropout or weight decay (Hernández-García and König, 2019).

Many different transformations can be applied to image data. Lighter augmentations include flipping an image either horizontally or vertically or translating the image by some vector. Heavier augmentations include more affine transformations, such as rotating, sheering, scaling the image, or adjusting the image's contrast, brightness, and hue.

Augmentations are limited only by the fact that they must preserve the necessary information that the model needs to make a prediction and by the computational cost they impose on the training procedure. In object detection, augmentations must also transform the ground truth labels, which also incurs additional costs.

2.2.6 Transfer learning

A different approach to solving for small datasets is the concept of *transfer learning*. There is a generally accepted assumption in machine learning that the training and testing data must be sampled from the same distribution and share the same feature space. Transfer learning challenges this assumption (Pan and Yang, 2010). A successful knowledge transfer can lead to a better generalization in a new domain with less data by training a model for one task in a domain with an abundance of data.

Transfer learning is widely used in the models presented in Chapter 3, both those used in pollen classification and general object detection. The source is usually an image classification model trained on a large dataset, such as the previously mentioned ResNet and AlexNet architectures. With object detection systems, the pre-trained model functions as a feature extractor for the detection architecture. In the domain of pollen classification, transfer learning has been shown to improve accuracy in CNN based classifiers, even though the source and target domain are vastly different.

2.3 Recurrent Neural Networks

Recurrent neural networks are not a major part of this thesis, but are used closely related work, so understanding the basic workings of this class of neural networks is needed.

A recurrent neural network is a special type of network used to process sequences of information, e.g., signals, text, or time-series. It adds *working memory* to the layers such that the activation of previous elements in a sequence are ‘remembered’. Given an input sequence $\mathbf{X} = \{x^1, \dots, x^n\}$, each element is activated in turn, but when processing element x^n, the activation of x^{n-1} is added.

2.4 Metrics

An essential step towards building a model is defining how to measure its performance. Implicitly, this is done through the construction of a *Loss function*. The models examined in this thesis do not employ novel Loss functions, so delving into their construction is not warranted. However, the metrics used when measuring the performance of object detectors, specifically, are of interest.

Object detection is a multi-task problem incorporating both the localization and classification of objects. Throughout this thesis, when referring to a *detection* made by an object detection system, this refers to a proposed boundary that the system believes encloses an object of a particular class. Every detection encodes both a localization and a class label. A correct detection refers to a detection that matches a ground truth, i.e., a predicted boundary with a particular class matches that of a ground truth of the same class.

2.4.1 Precision and recall

The precision and recall of a model refer to its ability to correctly locate and label the objects within an image. Before defining precision and recall, the following quantities must be introduced,

True Positive (TP): Number of objects correctly located and labelled.

False Positive (FP): Number of incorrect predictions.

True Negative (TN): Correct non-prediction, not usually relevant.

False Negative (FN): Number of objects missed by model.

Precision measures the model's accuracy, i.e., how many of the positive predictions are correct. Recall measures how many of the positive instances the model correctly labels. They are computed from the above quantities as follows,

$$\textbf{precision} = \frac{TP}{TP + FP}$$

$$\textbf{recall} = \frac{TP}{TP + FN}$$

These two metrics are the basis for how all object detection models are evaluated, and there is usually a tradeoff between the two. For instance, a model can have a very high recall, meaning it correctly identifies most potential objects, but the precision is reduced if it also identifies many other non-objects. Inversely, a model could be sure that it returns correctly identified objects at the cost of ignoring objects it is unsure about.

A popular accuracy measure that derives from the precision and recall values is the F_1 score, and it may also be referred to as the dice score. It is defined as follows,

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

The F_1 score measures the balance between precision and recall values and is useful in cases where both measures are *equally* important performance indicators.

2.4.2 Intersection over union

The correctness of a detection has been defined as 'a detection that matches a ground truth'. Classification has a simple binary solution; two classes are either the same or different. A positive solution to a binary localization problem would require pixel-perfect similarity between the predicted boundary and ground truth, which would be an extremely high bar to clear. For a more lenient approach, one could instead assign a *localization score* to the predicted boundary based on how well it matches the ground truth. A positive solution could then be defined as a boundary with a localization score above some threshold.

Most object detection systems use *intersection over union* (IoU) to score the match between boundaries. As the name indicates, it is defined as the ratio between the intersection and union of two boundaries,

$$\text{IoU} = \frac{\text{area of intersection}}{\text{area of union}}$$

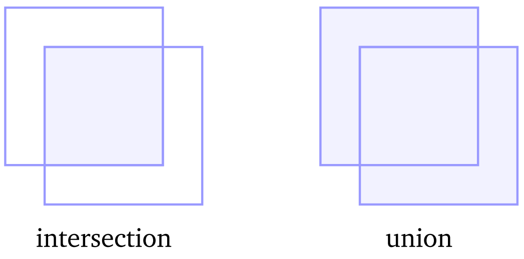


Figure 2.5: Visualization of the intersection over union of two boundaries. The named region is shaded.

The definition of what is considered a correct prediction (TP) can then be defined. Given prediction $\hat{X}$ with label $\hat{X}_l$ and bounding box $\hat{X}_u$. $\hat{X}$ is considered a True Positive if there exists a ground truth Y, where $Y_l = \hat{X}_l$ and $\text{IoU}(\hat{X}_u, Y_u) \geq \mu$. Where μ is some threshold value, often 0.5.

2.4.3 Mean average precision

Mean average precision (mAP) is a popular metric for measuring the performance of object detection models. It is computed by taking the mean of the *average precision* values for each class.

From a list of all detections made for a class, ranked in ascending order of confidence, each is labeled either true positive or false negative. In cases where multiple predictions match the same ground truth, only the highest-ranking prediction is considered a true positive. The cumulative precision and recall are computed from the ordered list of predictions, and from these values, a precision-recall curve is drawn. This shows how precision changes as recall rises over the range [0,1] as more and more detections from the ranked list are included in the precision/recall calculations. The AP describes the shape of the precision-recall curve and can be calculated in a few different ways. This thesis will use the definition of AP specified in the evaluation procedure for the VOC2007 image detection challenge. For convenience, the definition of AP, as given in Everingham et al. (2010) is repeated below.

AP is measured by taking the mean of precision values taken at 11 evenly spaced recall values as follows,

$$AP = \frac{1}{11} \sum_{r \in \{0, 0.1, \ldots, 1\}} p_{interp}(r) \tag{2.1}$$

Because the precision-recall curve often times is quite erratic, the precision value at a given recall level, r, is interpolated by finding the maximum precision value at any recall level exceeding r,

$$p_{interp}(r) = \max_{\tilde{r}:\tilde{r} \geq r} p(\tilde{r})$$

This section has detailed the current methods employed for automated pollen counting and the foundational building blocks of a CNN. Most research and application of CNN based methods is regarding classification, which only solves part of the problem of counting pollen. A subcategory of deep CNNs capable of predicting both classes and locations is required to automate the task fully. The next chapter will detail how *object detection* can be solved using a CNN by detailing how they have been used to solve tasks similar to pollen counting. The available literature relating to other attempts at solving the problem of counting pollen will also be given.

Chapter 3

Related Work

Object detection using CNNs is a relatively new area of study, and as such, its application in the domain of pollen counting lacks in literature. Therefore, examining related work requires a broader field of view and must explore how similar methods have been used to solve similar problems. This chapter is broken into two main sections. Section 3.1 will examine the various object detection frameworks and their use within microscopy. Section 3.2 will detail the various methods that have been employed with regards to the specific domain of automated pollen counting.

3.1 Convolutional Neural Networks

Before CNNs, the task of classifying images was usually highly dependent on the problem domain. Careful feature engineering was used to extract a set of parameters classified using a statistical model. A CNN fundamentally changes this landscape by removing all manual feature engineering. Over the last ten years, CNNs have risen to prominence as state-of-the-art in image processing. Raw images are classified directly, with little consideration of the specific domain. The trade-off is the nearly insatiable thirst these models have for labeled training data required to train them.

3.1.1 Object detection

With the quality of image classifiers rising, focus has been given to the more complex task of object detection, where the model must identify the location of objects within an image and their class. Work on this problem was kickstarted by Girshick et al. (2014) with the proposed method: *Regions with CNN features* (R-CNN). This three-stage system first identifies 'regions of interests' within an image before classifying them using a CNN and statistical classifier. They later proposed Fast R-CNN and Faster R-CNN, which improved the learning and inference time and the robustness of the original model.

R-CNN has three main modules. First, bounding boxes are proposed using selective search, an algorithm where different similarity measures are first used to segment the image into a myriad of small sections before these are then selectively grouped into larger *regions of interest*. Each region is then resized and fed into a CNN, the second stage, which produces a feature embedding which is finally classified using a Support Vector Machine (SVM), the third stage. An SVM is a supervised learning model for classification which attempts to maximize the margin between a decision boundary the training data (B. E. Boser, Guyon, and Vapnik, n.d.). A major computational bottleneck was having to process each region proposal independently through the second and third stages.

Fast R-CNN removes the second and third stages and replaces them with a new CNN, which considers both the whole image and the region proposals from the selective search. The CNN generates classifications for all region proposals with one forward pass, dramatically reducing the computational cost of this stage compared to R-CNN, where each region is classified in turn. This new second stage also predicts offsets for the proposed boxes, allowing it to refine the proposals from the first stage.

Faster R-CNN replaces the first stage with a Region Proposal Network (RPN), a fully convolutional deep neural network which produces a fixed number of bounding boxes together with an ‘objectness’ score for each box. The RPN introduces anchors, which are points in the image used to regress bounding boxes. After a set of convolutional layers, a $n \times n$ feature map is outputted. Imagining a set of anchor boxes imposed upon the image, which are centered on each cell of the feature map, the dimensions of regions of interest are computed by producing regressions of the anchor boxes by running a convolution over the feature map with a small kernel. At each step of the convolution, one parameter of an anchor box centered at the middle of the receptive field is produced. With four filters, the center point, height, and width of an anchor box can be regressed to a region of interest in proximity to the anchor. Using an RPN allows for training of both stages of the detector, significantly increasing the performance. Following the release of Faster R-CNN, the amount of research attempting to automate various object detection tasks has increased.

In many domains, there is usually a positive correlation between the cost of data and its quality. Often, methods that are proposed become prohibitively expensive because they use higher quality data only available to the researchers. CNN based methods have shown that high-quality models can be created using lower quality data. M. El-Melegy, Mohamed, and T. El-Melegy (2019) gave a good example of this. A Faster R-CNN method is proposed for detecting tuberculosis bacilli in LM slides. The proposed model can outperform all previous traditional models, many of which use higher quality imaging methods. The types of images the model uses are challenging to diagnose manually but are by far the most available in the field. The model also solves an issue present in most of the previous work, namely, how to automate diagnosis. Previous work uses pre-segmented images, which are then classified, requiring a human expert.

3.1.2 Single stage detectors

Common to the R-CNN family of methods is the use of two separate stages: one for identifying regions of interest in an image and classifying objects in those regions. This adds considerable complexity in that both systems require separate training and hyper-parameter tuning. These methods have been successfully utilized in many domains, including the only published attempt at pollen grain detection by Gallardo-Caballero et al. (2019). However, the inference speed is prohibitively slow for tasks that require real-time performance. Overall, there is a noticeable trend in the evolution of the two-stage systems where a CNN replaces stages. This trend continues to its logical conclusion with the development of the Single Stage Detector (SSD).

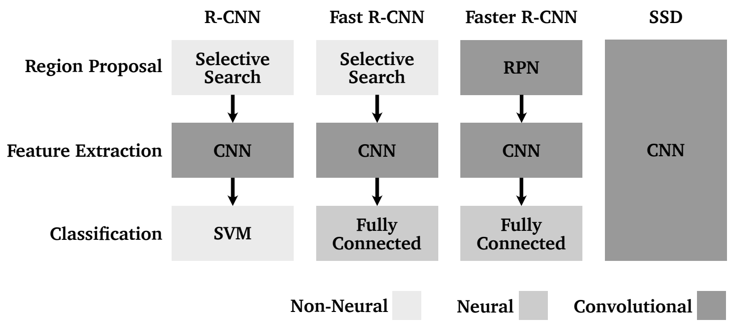


Figure 3.1: Object detection models show a gradual shift towards all tasks being performed by a CNN

This class of model comprises only a single CNN, responsible for both localizing and classifying objects. These models can be trained in one pass and feature inference speeds orders of magnitude faster than Faster R-CNN. One of the first methods was the Single Shot Multibox Detector (SSD), proposed by Wei Liu et al. (2016), which is the model implemented in Section 4.3. It was one of the first systems to demonstrate that a single fully convolutional model could vastly improve inference speeds without compromising accuracy.

The architecture resembles the RPN from Faster R-CNN. Like an RPN, SSD predicts bounding box offsets for a fixed number of *default bounding boxes*. Boxes with a predefined aspect ratio and size are centered on each cell of a feature map and a small-kernelled convolution predicts regression parameters for the position and size of these boxes. A separate but identically configured convolutional layer produces class confidence scores for each default box. Thus, the output of the network is both regression parameters and class confidence scores for a fixed number of default bounding boxes.

SSD makes predictions from multiple feature layers, and the default boxes are scaled up as the depth of the feature map in the network increases. This allows the network to predict objects at different scales in the image. The training objective for SSD is created by matching default boxes with overlapping ground truth boxes based on their IoU score. This match is then used to produce a target for the regression parameters and class for every positively matched default box. The model is trained by a linear combination of two separate loss functions, one for the bounding box regressions and one for the class confidences. Most default boxes are not matched to any ground truth box, which causes an imbalance in the loss function. To balance out the ratio of negative and positive examples in the training objective, only the negative examples with the highest loss values are included in the final loss calculation.

As of writing, there are no published attempts of using a single stage detector to count pollen grains, but there are examples of its use in similar domains. W. Liu, Cheng, and Meng (2018) uses an SSD model to detect brain slices used in an automatic sample preparation system. The model was chosen for its speed and accuracy, both important in real-time detection. The model was simplified with a smaller range of default box scales because the domain is less varied than the training data for the standard SSD model. This simplification could also prove relevant to a pollen detection task where grains are similar in size and shape. Because of SSD's structure, this simplification is achieved by removing layers from the model, thus removing predictions from those scales. Results show that the simplified model increased both accuracy and speed over the original.

You Only Look Once (YOLO) is a very popular single stage model, its release closely following SSD (Redmon et al., 2016). As with R-CNN it has been released in many versions with iterative improvements. It is similar to SSD in that it also predicts box offsets and class scores for a fixed number of bounding boxes. Specifically, YOLO divides an input image into a grid and then predicts box offsets for a fixed number of bounding boxes centered in each grid square (but not regressed from default boxes) and class confidences for each grid square (not each bounding box). Both SSD and YOLO use transfer learning in the form of an initial feature extractor transferred from a pertained classification model. The later versions of YOLO also feature multiple extraction layers, which improve predictions for smaller objects, which was one of the significant weaknesses of the initial version.

Recently, multiple papers using YOLO models have been published showing promising results in areas similar to pollen counting. Chibuta and Acar (2020) used a modified version of the third iteration of YOLO to screen blood smears for malaria. Diagnosing malaria is very costly because it requires manual analysis of blood samples. As with Tuberculosis, the areas with the highest prevalence of the disease are those where the primary screening technique uses light field microscopy. The presented model uses a smaller feature extractor and fewer extraction layers to optimize for speed on basic hardware. The model has $\simeq 99\%$ fewer parameters than the standard YOLOv3 implementation and still performs at the same level as human experts and its unoptimized equivalent.

Another area of study of relevance to this thesis is blood cell counting. A complete blood count is a test that is often requested when evaluating general health and involves a manual count of blood cells within a sample. The current standard process requires human expert analysis and is prone to error. Islam and Alam

(2019) proposed a YOLO model which accurately localizes and classifies blood cells using standard LM images of a blood smear. The YOLO model has not been modified apart from changing the number of classes to 3 (Red, White, and Platelets). However, they do change the inference routine to optimize the count for each of the three cell types. As with the aforementioned RPN, YOLO predicts an objectness score for each bounding box and usually considers a box to be a positive match if the score exceeds a threshold. Islam and Alam show that, rather than using only one threshold value for all classes, higher overall accuracy can be reached by filtering boxes for each class independently with different thresholds. This points towards a larger issue of choosing the algorithms used to filter the predictions made by these models and their hyperparameters. The difference between how many raw predictions a model makes and the number of objects an image contains is usually many orders of magnitude. The method used to filter the predictions is therefore crucial to the model's performance and highly dependent on the domain.

3.2 Automated Pollen Detection

There have been many attempts at pollen grain classification over the last three decades. These have been nicely summarized by Sevillano and Aznarte (2018). Most are statistical classifiers using selected features from pollen images. The earlier attempts can be grouped into three categories. The first focuses on morphological features such as shape, size, and symmetry. The second type uses the texture of the grain surface as the discriminating feature. The last group uses a hybrid approach that combines morphological and texture features. These methods have successfully classified pollen to a degree comparable to human experts, but all rely on careful feature engineering. Of the earlier methods, the most successful utilize images taken through SEM, which is a much more expensive imaging technique than standard LM imaging.

3.2.1 Classical methods

As a precursor to the newer systems, it is pertinent to cover the earlier attempts at solving the task of automated pollen counting. The first attempt, by Langford, Taylor, and Flenley (1990), used greyscale SEM images of the surface texture on pollen grains. Based on a Grey-tone spatial dependence analysis, six feature measures were then produced and classified using Linear Discriminant Analysis (LDA). This technique was successful but required manual analysis for each class, making it challenging to apply to new datasets or other pollen taxa.

Other attempts were made over the next decade, some using morphological features instead of surface texture, but they follow the same basic procedure of feature engineering followed by a statistical classifier. The next significant contribution was made in Li and Flenley (1999), in which very high accuracy was achieved using LM images. The major disadvantage of LM imaging was the shallow depth of field, causing only portions of the pollen grains to be in focus.

This reduction of image quality caused a loss in the accuracy of the LDA-based methods. The new method exchanged the classifier with a Multi-Layer Perceptron and achieved higher scores than previous methods using a simpler feature measure. The main limitation was the lack of processing power at the time, which meant that the method could not scale to larger sets of images.

As computational power rose, there were also attempts made at localizing grains. France et al. (2000) presented a hybrid solution featuring both classical and neural methods. The localization is handled by a K-means classifier coupled with a shape and size filter, producing segments of the image likely to contain pollen. A trained classifier is then used to classify the grains. The results were promising, but the system was also very limited. Firstly, it was very sensitive to focus and could only work with grains perfectly within the depth of field. Secondly, the segmentation algorithm only worked on sparse images with a certain amount of space between grains. These same issues also create problems for modern systems, albeit not to the same extent.

Convolution has been an important tool in image processing since before CNNs gained traction. Using hand-crafted filters, many essential features such as edges can be extracted from an image. This technique was employed by Amar Daood, Eraldo Ribeiro, and Mark Bush (2016) with good results. The system used an SVM for the final classification but demonstrated the viability of using convolutional filters as feature extractors. In some ways, the system bridges the gap towards the CNN models but crucially lacks the ability the learn which features should be extracted. This is the fundamental deficiency common to all the classical methods: they rely on human expert knowledge to adapt each method for use in the specific domain.

3.2.2 CNN methods

CNNs have taken over as the standard in image classification, and this is also the case in pollen detection. Recently all the proposed methods involve a deep convolutional neural network as the primary feature extractor. Comparing the different models that have been presented is very challenging. Most use self-collected datasets that vary in size, both with respect to the number of classes and examples per class. There is also an inherent difference in the difficulty of separating instances within any given dataset because some types of pollen are much more similar than others. Meaningful comparisons are therefore challenging to make. However, a performance comparison is not strictly relevant to this thesis, seeing as they are all classifiers, which cannot be used to locate pollen grains.

A. Daood, E. Ribeiro, and M. Bush (2016) presented a CNN method that was used on both an LM and SEM dataset and compared the results with many of the classic statistical methods, showing the clear benefit of using a CNN model. A second network was also implemented that used transfer learning to improve accuracy further. Data augmentation was used to combat the small size of the dataset, which has been a pervading issue for most of the presented solutions. Although a higher accuracy was obtained on the SEM data, the paper showed that the models were fully capable of achieving good results using LM images. Both transfer learning and data augmentation are featured in most of the subsequently published papers.

Sevillano and Aznarte (2018) later gave more evidence for the supremacy of CNN methods by applying three different convolutional models on a publicly available dataset POLENE23E, which at that point only had been classified using classic methods. All three models used a CNN as the feature extractor. They all performed well, doubling the precision over the state-of-the-art. The results also show that there are only insignificant improvements when using a linear discriminant classifier on top of the CNN.

Common to all the mentioned methods is that the data they rely on is LM images of a single grain. This ignores two important factors. Firstly and most obvious is that none of these models can be used directly to count pollen grains, only to classify pre-segmented images of singular grains. This also leads into the second important factor of dealing with LM images of pollen grains at different focal planes.

Grains of pollen on a slide are not distributed across one focal plane and are not oriented in the same direction. Grains are scattered across all three axes and, because of the narrow depth of field of the microscope, are only partially visible when in focus. The depth of field is so shallow that only parts of the surface ornamentation appear clear. This is not a problem for a human operator as the focus can be adjusted to reveal the entire grain. However, with static images, dealing with this lack of complete information is an open question. Figure 3.2 shows an example of this and shows a grain observed at three different focal planes.

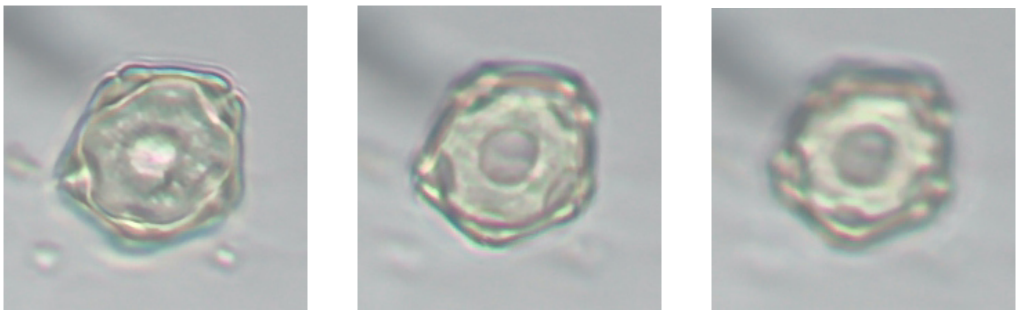


Figure 3.2: z-stack of a pollen grain taken at three focal planes. The focal plane moves down from the top (left) to the bottom (right). With finer adjustments, many more images can be produced from a single grain.

For the models we have covered, this means that they potentially are missing out on features because they only rely on an image of a single focal plane. In the case of classification, one possible remedy for this is to use a stack of images, named a *z-stack*, taken at different focal planes and process the stack as one unit. Amar Daood, Eraldo Ribeiro, and Mark Bush (2018) uses this approach. The model they propose takes as input a sequence of 10 images taken of pollen grains spanning the whole grain and classifies it using a network that combines a recurrent neural network (RNN) and a convolutional neural network. The model first uses a CNN to create a feature embedding for each frame independently, and an RNN then classifies the sequence. The result is a reportedly 100% accurate model over a dataset with ten classes.

The main conclusion is that the z-stack benefits from being processed as a sequence and not as independent samples. However, the results show only minor improvements from using the RNN over a majority-vote system using classifications from the CNN directly. There is also no clear strategy for extending the system for grain detection, which limits its utility.

The last model we will look at is the only published work attempting to use a CNN to localize pollen grains in slide images. Gallardo-Caballero et al. (2019) uses Faster R-CNN to detect pollen grains of various types within unaltered LM images of pollen grains. The model does not classify the grains it localizes. They report very high values for precision and recall but use a slightly modified definition of IoU when calculating these values. Multifocal data is only used when running inference by combining the detections from the entire z-stack as one single prediction.

The dataset they use was created by filming the slide while moving the focal plane across the pollen grains. Based on an auto-focused keyframe, ten frames were extracted before and after the keyframe, creating a 21-frame z-stack. From this, two datasets were created. In the first one, the pollen grains were labeled in the frame where they appeared sharpest and ignored in all other images. In the second dataset, the grains were labeled in all images throughout the sequence.

The model's performance was calculated by stacking the individual predictions from all focal planes together and then performing a standard filtering algorithm. The reported results are excellent, but the results are probably inflated due to the elimination of class predictions from the problem. The definition of IoU is also changed so that comparison to other object detection models is impossible. The effect this has on the values for recall and precision is not declared.

Two trials were run to compare the performance obtained from the two datasets, and values for recall and precision were high for both (above .98), with the non-blurred trial having slightly higher precision. The authors conclude that there do not seem to be any significant differences in performance between models trained with or without blurred images. However, their method of using detections from multiple focal planes in each prediction hides the effects that blur has on the model hard to pinpoint.

How to utilize the information contained in a z-stack is a very open question. Both methods presented above using z-stacks achieved good results, but neither attempts to explain how the models are affected by the different sharpness valued in the data. Instead, they aggregate information from all images, which hides how the model responds to each static image. Gathering data on how sharpness affects the model could provide valuable insight into this relationship.

This section has detailed how CNNs are used to solve object detection problems and how the field of automated pollen grain analysis has evolved from using highly specialized hand-crafted feature extractors to being dominated by generalized CNN based frameworks. Of most significance to this research are the modifications that have been successfully made to CNN models used in similar fields to pollen grain counting and the different techniques that have been used to make use of multifocal data.

Chapter 4

Methodology

The goals of this research require the development of several components, chiefly among them a pollen dataset and object detection model. Section 4.1 covers the data collection and Section 4.3 details the model architecture. As a prerequisite for being able to analyze sharpness, an objective sharpness measure is needed. A theoretical background for this is given in Section 4.2. Finally, the experimental setup as well as the experiments themselves are explained in Sections 4.4 & 4.5, respectively.

4.1 Data

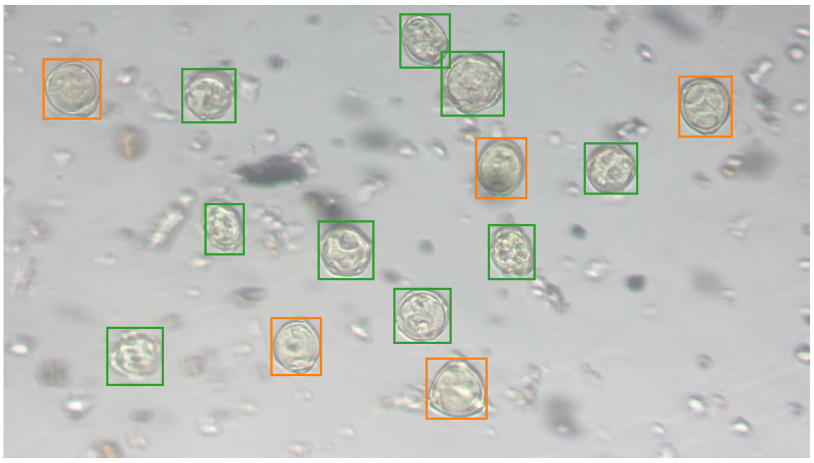


Figure 4.1: Example from the dataset with ground truth bounding boxes drawn. The image contains two classes: corylus, and alnus.

The results presented in Chapter 5 are trained on data sourced from the Norwegian Asthma and Allergy Association, which has, since 1980, tracked the amount of airborn pollen in Norway. Pollen is collected with traps where the air is continually sucked through a small slit and is redirected over an adhesive strip. The strip is moved across the slit, exposing different sections throughout the day. Pollen grains and other air-born particulates adhere to the strip, which is then analyzed under a microscope. Only pollen grains from a subset of species are actively tracked.

Three microscope slides have been imaged using a digital optical microscope producing a set of 701 raster images with a size of 1080×1920 pixels and three channels; red, green, and blue. The resolution of each image is $0.183\,\mu\mathrm{m}\,\mathrm{pixel}^{-1}$. Each image has been labeled in collaboration with the experts to produce valid and correct ground truths. In total, three different species have been classified, namely *poaceae*, *corylus*, and *alnus*, known in English as Grasses, Hazel, and Alder, respectively. A labeled example is given in Figure 4.1. A summary of the dataset is given in Table 4.1.

Table 4.1: Distribution of class labels across the 701 sample images of the pollen dataset.

	Poaceae	Corylus	Alnus
Number of labels	5600	262	522
Proportion	87.7%	4.1%	8.2%

Many images are taken from the same viewpoint but with the focal plane set to different grains. The ground truth labels are drawn for all present pollen grains, regardless of how blurred they appear. This is done so that the dataset may be modified to analyze sharpness and model performance in regards to *RQ2*.

As opposed to more general object detection tasks, where there is a significant variance in both the apparent size and shape of objects within an image, this dataset is much more regular. Looking at Figure 4.2, the grains are mostly circular and between 100 and 150 pixels wide.

4.2 Sharpness Measure

Analyzing how the sharpness of pollen grains affects detection performance requires an objective sharpness measure. This section details the chosen method of measuring the local sharpness of pollen grains within sample images. The measure is based on Fourier analysis, and its performance has been tested on the training data.

4.2.1 Fourier analysis

Fourier analysis describes the general method of utilizing the Fourier transform to analyze the component frequencies present in some signal. For the purpose of Fourier analysis, imagine the image as a collection of signals, each of which describes the change in brightness value when traveling across the image in some direction. For an $M \times N$ image, the two-dimensional Discrete Fourier Transform is defined as follows,

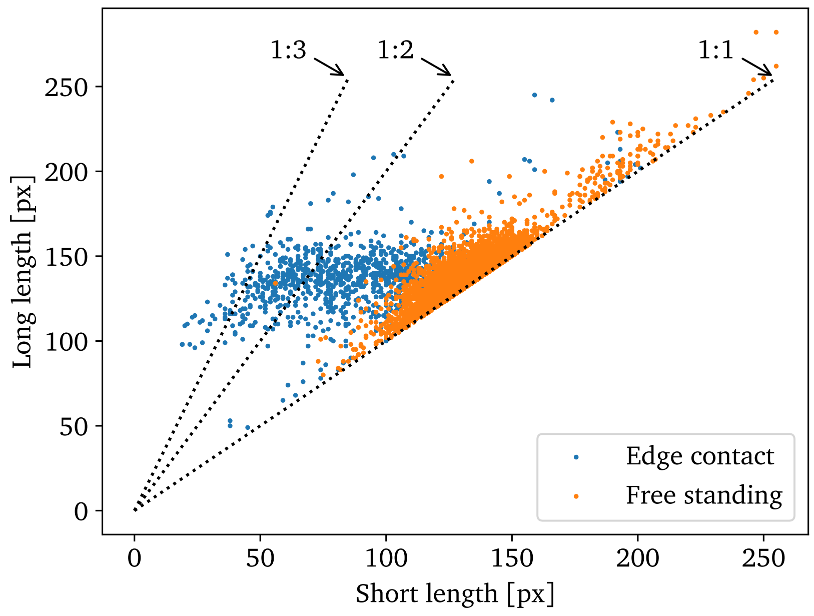


Figure 4.2: The dimensions of all ground truth boxes are plotted, longest against shortest. The lines denote the three aspect ratios used in the default boxes of the SSD model. Grains marked 'Edge contact' are in direct contact with the edge of the image and are most likely partially cropped out of frame. The grains are all quite regular in both shape and size

$$F(u,v) = \sum_{m=0}^{M}\sum_{n=0}^{N} f(m,n)\, e^{-i2\pi\left(u\frac{m}{M}+v\frac{n}{N}\right)} \tag{4.1}$$

$$e^{-i2\pi(ux+vy)} = \cos 2\pi(ux+vy) + i \sin 2\pi(ux+vy)$$

where $f(m,n)$ is the spatial domain of the image, and the exponential term is the basis function at each point of $F(u,v)$ in the Fourier domain. Taking the Fourier transform of an image then produces a 2-dimensional matrix where the intensity of each element represents a the coefficient of a 2-dimensional sinusoid basis function of the image. Figure 4.3 visualizes what the basis functions can look like and demonstrates exactly what is encoded in the Fourier spectrum.

Figure 4.4 shows the Fourier transform of various inputs. The transforms are shifted, such that $F(0,0)$ is in the center of the transform. The maximum frequency representable in the spatial domain is a 2-pixel wide stripe pattern going from minimum to maximum brightness. u and v represent the number of oscillations in each direction of the basis function, so the maximum frequency gives $M/2$ and $N/2$ oscillations, respectively. The directionality of the Fourier transform is demonstrated in the first two examples. Squares decompose into a set of sinusoids, all moving in the same two directions. The coefficient of each component is

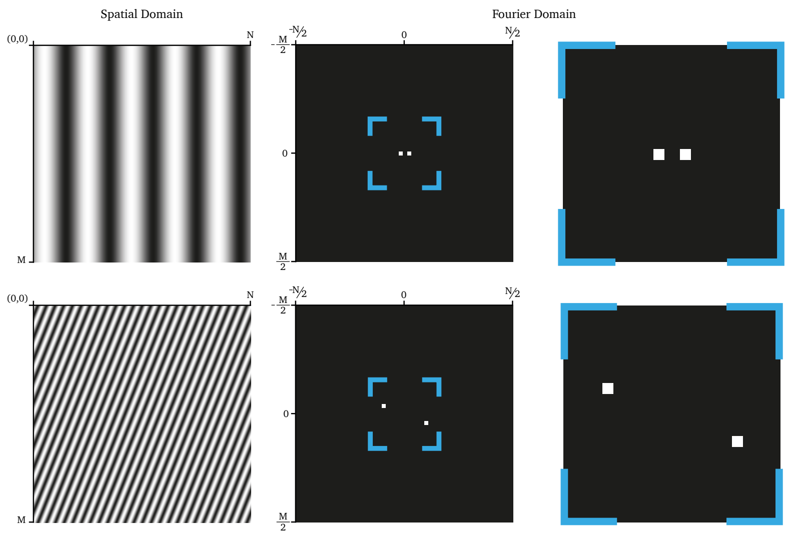


Figure 4.3: The figure shows two 2D sinusoid basis functions and their Fourier transform. The active components in the transform have been enlarged to make them visible in print. When unmodified, only a single pixel and its reflection about the origin is active.

encoded in the intensity of the pixel. In all the examples, the lower frequency components dominate, which lights up the center region. The last two examples demonstrate how blurring an image eliminates the higher frequencies from the Fourier domain.

Using the Fourier spectrum to measure sharpness follows from the realization that a strong relationship exists between the sharpness of an image in the spatial domain and the distribution of frequency components in the frequency domain. Sharp features produce high frequencies while blur smooths out the changes in brightness, lowering the frequencies. Figure 4.5 shows three different pollen grains, captured with progressively more blur. By visual inspection, it is clear that as the perceived blur increases, high-frequency components also decrease.

4.2.2 Measuring sharpness

The problem then is to decide how to encode this change in frequency distribution as a scalar sharpness measure. De and Masilamani propose a simple method that counts the number of components in the Fourier spectrum having a value above a certain threshold. The operation is described in Equation (4.2).

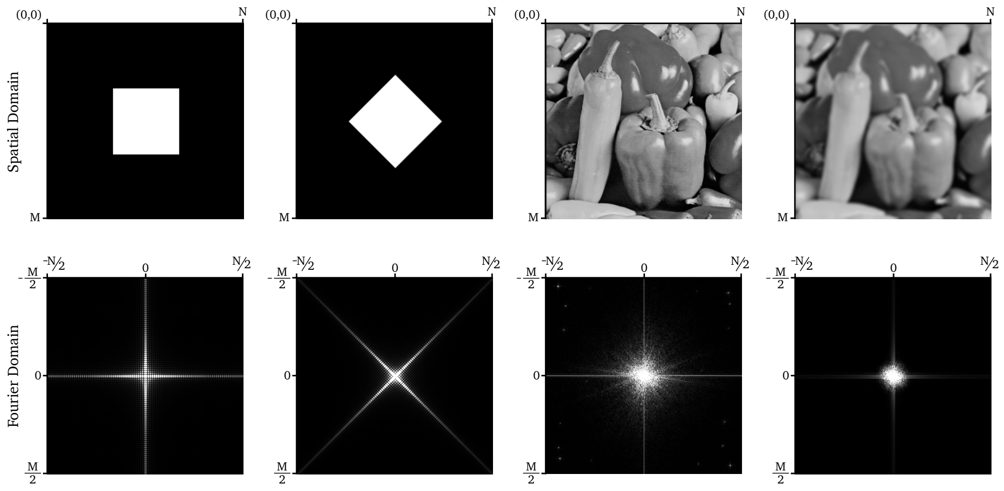


Figure 4.4: Demonstration of the Fourier transform. The bottom row shows the center-shifted discrete Fourier transform of the corresponding image in the top row. The Fourier spectra mapped to grayscale values with a truncated linear amplitude mapping which is scaled to compensate for the otherwise dominating center component.

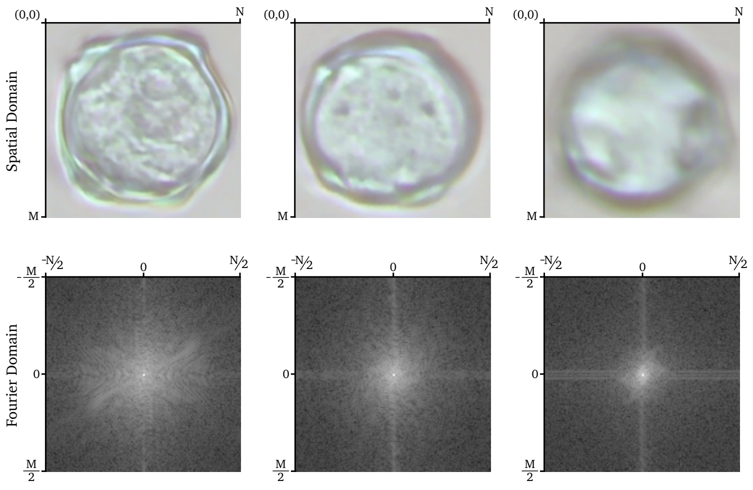


Figure 4.5: Pollen grains and their corresponding centered Fourier spectrum. The Fourier spectra are log scaled so that the higher frequencies become visible.

$$\begin{aligned}
\mathbf{X} &= \mathcal{F}(a), \quad a \in \mathbb{Z}^{M \times N} \\
T_H &= \sum_{x \in \mathbf{X}} [x \geq \mu], \quad \mu = \frac{\max \mathbf{X}}{1000} \\
S &= \frac{T_H}{M \cdot N}
\end{aligned} \tag{4.2}$$

Here $\mathcal{F}$ is the discrete Fourier transform operating on an input image a, which is a $M \times N$ matrix of integers. The phase component of the complex-valued output is ignored, leaving $\mathbf{X}$ to contain only the magnitude of the frequency components. The scaling factor of the threshold value, μ, was found to produce good results on the data without modification. S is the sharpness measure.

Validating the sharpness measure is an important task. The weight of any argument made based on analysis using this measure is predicated on its soundness. There are many different approaches to this, one of which is to compare the objective measure with subjective measurements of perceived sharpness on a subset of the training data.

4.2.3 Evaluating the sharpness measure

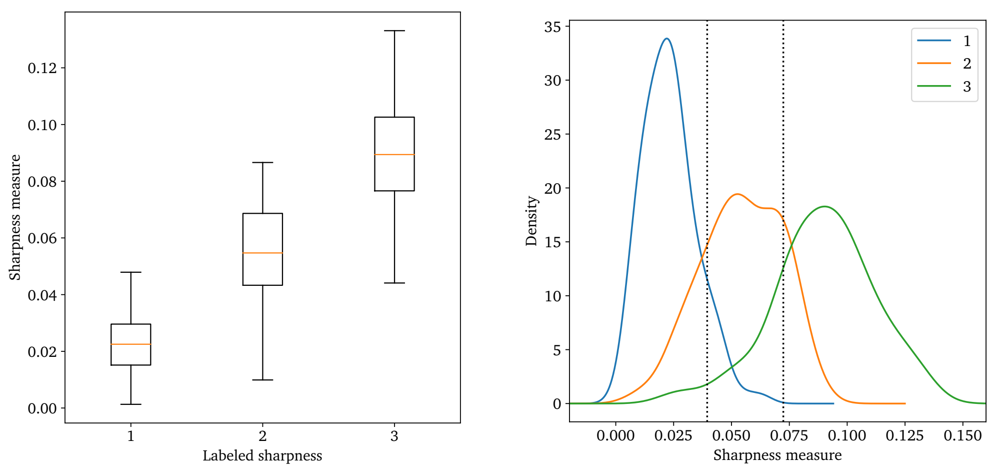
The basis of the evaluation is a new dataset created from a small random subset of the training examples. Each ground truth is then given sharpness scores based on perceived sharpness and the objective measure. Determining perceived sharpness of a pollen grain with high fidelity was found to be highly subjective and non-reproducible with repeated independent scoring, so a simple classification was instead performed. Images were separated into three classes representing perceived sharpness. Figure 4.5 gives examples of the classes, with the top left image being the sharpest (class 3) and the top right image being the blurriest (class 1). A total of 316 pollen grains were evaluated; the distribution of their classes is given in Table 4.2

Table 4.2: Distribution of classes across the sharpness evaluation dataset.

	blurry [1]	partly [2]	sharp [3]
Number of labels	108	93	115
Share	.34	.30	.36

Figure 4.6 shows a clear correlation between the mean of the distribution and the perceived sharpness. The overlap is to be expected, given the mapping from a continuous predictor onto a categorical label. To further evaluate the performance of the sharpness measure, a very simple decision tree model is constructed, and its ability to differentiate between the classes is tested. The model achieved a test accuracy of 83.5%.

Figure 4.7 shows the computed the sharpness of every ground truth in the dataset in a density plot, visualizing the approximate distribution. As expected, most



(a) Box plot **(b)** Density plot

Figure 4.6: The sharpness measure grouped by class label. There is a large overlap between adjacent classes, but the IQRs in (a) are clearly separated. The correlation between perceived and measured sharpness is also clear.

ground truths fall into the sharpest category, but there is a spread, allowing for the analysis of the relationship between sharpness and model performance.

4.3 Architecture

The model that has been implemented is the Single Shot Multibox Detector, a brief explanation of which was given in Section 3.1.2. This section gives a more thorough explanation of the model and its implementation, focusing on the changes made from the original. Most of these changes are motivated by more recent research featured in newer models. The reasoning behind choosing a relatively old model framework is based chiefly on its architecture, which lends itself to alteration and simplification.

The implementation is based on an implementation by NVIDIA[1], the basic structure is shown in Figure 4.8. The implementation can be divided into four parts. A feature extraction network first transforms the input image into a set of high-level feature maps, which are then fed into an *auxiliary structure* which gradually steps down the dimensions of the feature map and, in the process, creates a set of source feature maps. Two detection structures then consume the source feature maps, which produces the final predictions. One set of filters produces class confidence scores, and one produces bounding box regressions.

[1]Released under the Apache 2.0 License, URL:
`https://github.com/NVIDIA/DeepLearningExamples/tree/master/PyTorch/Detection/SSD`

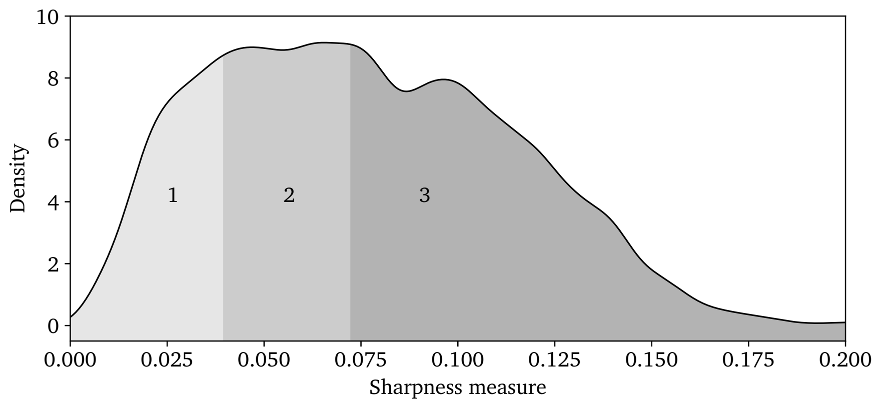


Figure 4.7: Density plot showing the sharpness of all pollen grains in the dataset ($N = 6384$). The shaded sections indicate the distribution of sharpness classes if the dataset is classified with the evaluation model. As expected, most grains fall into the sharpest category.

4.3.1 Model

The model comprises three distinct parts, a feature extraction network transferred from a pre-trained model, an auxiliary structure that creates a set of feature maps, and a detection structure that creates the bounding box regressions and class predictions. The model is purely convolutional with no fully connected layers.

The feature extractor can be any classification network that can produce a feature map of the correct dimensions. In the original paper, the VGG-16 network is used (Simonyan and Zisserman, 2015) without any particular justification as to why. ResNet-34 is used in the baseline configuration. The main reason for this is its superior performance over VGG and the ease with which it integrates with the auxiliary structure (He et al., 2015). ResNet-34 is the largest of the ResNet family that fits on the GPU used for most of the experimentation. PyTorch offers implementations of the most popular pre-trained models as part of their API.

The auxiliary structure is implemented as a sequence of blocks where the output of each block is a feature map consumed by the detection structure. Each block contains two convolutional layers, as specified in Figure 4.8. Batch normalization is used after each convolution, and ReLU is used as the activation function.

The detection structure is implemented as a sequence of distinct layers, each of which runs over one of the intermediate feature maps from the auxiliary structure. Each localizing layer has four filters per default box, one for each of the regression parameters. Similarly, the class prediction layers have C filters per box, one for each class with an additional layer for the background.

When executing the forward pass through the model, the input is first passed through the feature extractor. It is then passed through each block of the auxiliary structure, saving each feature map. The final output is then generated by running

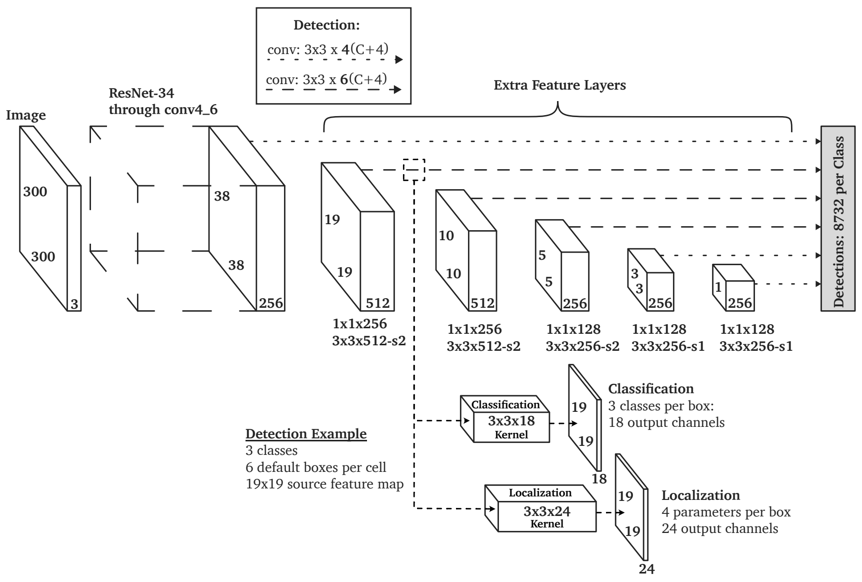


Figure 4.8: Overview of the model, inspired by the figure given in the original paper. However, it has been modified to reflect the alterations made and highlight how bounding box and class predictions are calculated with convolution. The horizontal lines denote the detection structure comprised of 3 × 3 convolutions for class confidences and bounding box regressions. C is the number of classes.

each feature map through the corresponding localization and classification block. All the outputs from the localization and classification layers are concatenated, producing two output tensors.

4.3.2 Default boxes

The fundamental objective of the model is to evaluate a predefined set of default boxes. Each location of a feature map forms the basis for a small number of default boxes of fixes size and aspect ratio. In Figure 4.9 the four default boxes shown are evaluated when the receptive field is centered on the marked square. The default boxes are scaled independently from the size of the feature map, but there is a correlation. Because of the depth of the feature extraction network leading into the feature layers, the receptive field of all elements in every feature map is the entire input. As designed, the SSD model is made to handle a wider distribution of sizes than is present in the pollen dataset. The sizes of the default boxes have been adjusted so that they cover the distribution of sizes present. Even with adjustments, the range of default box sizes is so large that it is questionable if all six feature maps are necessary.

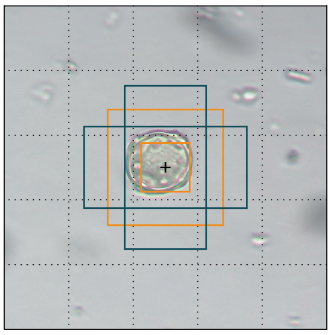


Figure 4.9: 4 default boxes on a 5 × 5 feature map.

4.3.3 Training objective

Because of the default boxes, each ground truth box needs to be matched to one or more default boxes so that target values for the regression and confidences can be created for the loss function. For each training example, a matching strategy is applied that finds default boxes that overlap with each ground truth.

SSD has a generous matching strategy where ground truths and default boxes are matched if their IoU is above a certain threshold. Figure 4.10 shows the result of the matching strategy being applied to a training example. Every red box in the image is a default box that has been matched with a ground truth, shown in green. For the purposes of training, these are the default boxes that the loss function expects the model to have correctly labeled and regressed. This ensures that there are multiple positive examples for each ground truth.

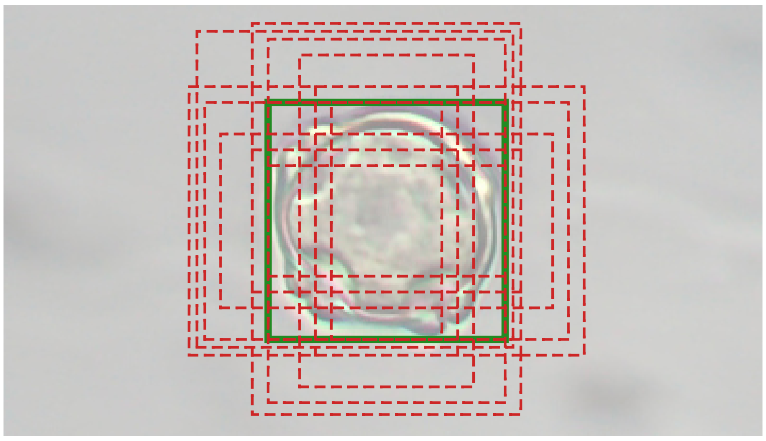


Figure 4.10: The result of the matching procedure. All default boxes which are matched to the ground truth are drawn with dashed red lines. Here, 19 default boxes are matched to the ground truth (`IoU` ≥ 0.5). This example has been handcrafted for clarity by setting the bounding box equal to a default box, causing a more symmetrical match.

The loss function, as given by Wei Liu et al. (2016), is calculated as follows,

$$L(x,c,l,g) = \frac{1}{N}\left(L_{conf}(x,c) + \alpha L_{loc}(x,l,g)\right) \tag{4.3}$$

where $x = \{1,0\}$ is a binary mask denoting a match between a default box (d) and ground truth (g), N is the number of positive examples, in the case of $N = 0$ the loss is set to 0. α is a weight term set to 1, showing little to no effect on performance. Here, *positive example* refers to default boxes which are matched to a ground truth, Figure 4.10 shows an example with $N = 19$. L_{loc} is the summed Smooth L1 loss between predicted boxes (l) and ground truth parameters (g), while L_{conf} is the SoftMax loss over multiple class confidences (c), both are standard loss functions provided by PyTorch.

As the target for the Smooth L1 loss, $\hat{g}$ defines the regression from the center (cx, cy) of the default box (d) and its height (h) and width (w) to a matched ground truth.

$$\hat{g}^{cx} = (g^{cx} - d^{cx})/d^{w} \qquad \hat{g}^{cy} = (g^{cy} - d^{cy})/d^{h}$$
$$\hat{g}^{w} = \log\left(\frac{g^{w}}{d^{w}}\right) \qquad \hat{g}^{h} = \log\left(\frac{g^{h}}{d^{h}}\right)$$

A problem that this and many similar models face is the gross imbalance between positive and negative training examples. Out of the 8732 default boxes, only a small subset can be matched to any given ground truth. For the localization loss, this has no impact as only the matched boxes are counted towards the loss. However, for the confidence loss, the overwhelming majority of predictions are negative matches (matched to background). To counteract this imbalance, SSD uses *hard negative mining* to balance out the negative and positive examples. As the name suggests, this technique aims to focus training on particularly 'hard' examples and ignore the others. All negative training examples are sorted by the confidence loss, which ranks them to how badly the model miss-classified them. Negative examples are counted towards the confidence loss by descending loss until the ratio between the number of positive and negative examples included in the total loss is 1:3.

4.3.4 Inference

When the model is working, the raw output it produces is regression values for many thousands of bounding boxes with class confidence predictions. The model will most likely produce multiple overlapping bounding boxes in the locations where an object is present. Ideally, the output would be a one-to-one mapping between the predicted boxes and objects in the image.

A common filtering technique is called *Non-maximum Suppression* (NMS). Let $\mathcal{B}$ be a list of bounding boxes and $\mathcal{S}$ be their corresponding scores. NMS is then a simple iterative filtering algorithm where, for each iteration, the bounding box with the highest score is selected from $\mathcal{B}$. Any bounding boxes that overlap with the selected box are then pruned from $\mathcal{B}$ and $\mathcal{S}$, and the loop is continued until $\mathcal{B}$ is exhausted. Boxes are considered overlapping if their IoU is more than some threshold N_t.

NMS performs well in most settings but has a particular weakness when the target boxes of objects overlap. In these situations, NMS will often only return a single bounding box for one object, ignoring the rest. Given that the data in this project exhibits this characteristic, the standard NMS algorithm is replaced with a more modern version, Soft-NMS, introduced in Bodla et al. (2017). The pseudocode for both algorithms is as follows,

```
def NMS($\mathcal{B} = \{b_1, \dots, b_N\}$, $\mathcal{S} = \{s_1, \dots, s_N\}$, $N_t$):
  $\mathcal{D} \leftarrow \{\}$
  while $\mathcal{B} \neq \emptyset$:
    $m \leftarrow \text{argmax } \mathcal{S}$
    $\mathcal{M} \leftarrow b_m$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{M}$; $\mathcal{B} \leftarrow \mathcal{B} - \mathcal{M}$
    for $b_i \in \mathcal{B}$:
      if IoU($\mathcal{M}$, $b_i$) $\geq N_t$:
        $\mathcal{B} \leftarrow \mathcal{B} - b_i$; $\mathcal{S} \leftarrow \mathcal{S} - s_i$          NMS

      $s_i \leftarrow s_i$f(IoU($\mathcal{M}$, $b_i$))  Soft-NMS
  return $\mathcal{D}$, $\mathcal{S}$
```

Soft-NMS changes only the colored sections, where it introduces a rescoring function, `f(IoU)`, which only decays the scores of remaining boxes instead of discarding them altogether. This allows boxes that overlap with the selected box to remain in the pool. The rescoring function `f(IoU)` applies a Gaussian decay as follows,

$$f(\gamma) = e^{-\frac{\gamma^2}{\sigma}}$$

where γ is the IoU as given in the pseudocode and σ controls the rate of decay, as demonstrated in Figure 4.11. $\sigma = 0.15$ was found to produce the best results.

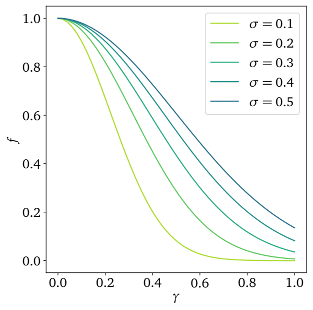


Figure 4.11: Gaussian decay as a function of IoU for various σ

4.4 Experimental setup

This section will present the experiments which have been designed in order to examine the research questions. Firstly, the software and hardware stack used in the implementation of the model is presented, together with the training procedure used for the experiments. A run-down of the experiments that have been performed will then be given.

4.4.1 Software

All the software for this project is written in Python, which is among the most established languages for deep learning. This project relies entirely on the extensive use of the high-quality open-source data science ecosystem created within Python. The model implementation, training, and evaluation procedures are written using the 1.7 release of the PyTorch library (Paszke et al., 2019). PyTorch is an extensive machine learning framework and provides APIs covering the entire model pipeline.

4.4.2 Hardware

All experiments have been run on a desktop computer using consumer grade components. The relevant specifications are as follows,

- **CPU:** *AMD Ryzen 5 3600X*, 6 cores @ 3.8 GHz.
- **GPU:** *Nvidia GTX 1070*, 8 GB GDDR5 VRAM,
  1920 CUDA cores @ 1506 MHz, 6.463 TFLOPS (32 bit).
- **Memory:** *Corsair Vengeance*, 32 GB DDR4 @ 2166 MHz.

With a batch size of 32, each forward/backward pass requires approximately 410 ms to complete. Adding the time to read data from disk and run augmentations, one full iteration takes 812 ms, bringing the total time for one experimental run of 2000 iterations to 27 minutes. The final model produces detections at a rate of 182 FPS, measuring only inference time.

4.4.3 Training

All the experiments were run using the same training setup. The model is trained using Stochastic Gradient Decent (SGD) with a mini batch size of 32, learning rate of 10^{-3}, 0.9 momentum, and 5×10^{-4} weight decay. In testing, test mAP converged after 2000 iterations in the baseline configuration.

The dataset contains three classes, but these are not equally distributed between the 701 images. Corylus and Alnus both release together and are found together in samples. Poaceae appears as the only class in the samples where it is present; it is also much more prevalent than the other species, meaning the density of pollen grains is much higher on average in these samples. This also causes the very uneven distribution of classes in the dataset. The Poaceae class is therefore under-sampled when creating the final dataset. 467 samples are included in the final dataset; the class distribution is given in Table 4.3. The final dataset is then divided into a training and test split with an 85% / 15% ratio.

Augmentations are necessary to allow the model to train for many enough iterations without overfitting on the training set. Augmentations artificially increase the size of the dataset, by producing thousands of variations of each example. The following augmentations are employed,

1. Horizontal flip.
2. Vertical flip.
3. Color adjustment (brightness, hue, saturation, contrast).
4. Shuffle RBG channels.
5. Random crop.

All the augmentations except cropping, independently, have a 50% chance of being applied to each sample. The random crop also resizes the sample input into a 300×300 square, so it is needed for every image. The cropping algorithm also ensures that each sample contains at least one pollen grain.

Table 4.3: Distribution of classes across the training dataset

	Poaceae	Corylus	Alnus
Number of labels	518	262	522
Share	.40	.20	.40

4.5 Experiments

Four experiments are designed, two targeting each research question. With the goal of developing a system for automated pollen counting, the aim is to explore how the peculiarities of this domain interact with the more established methods that have been developed for object detection. For each experiment, its rationale and setup are given.

4.5.1 Feature extractor

The largest section of the model is the feature extraction block, so with the goal of decreasing computational complexity, shrinking this block would have the most significant impact. Because this dataset differs so much from the common datasets used to pre-train the feature extractors, it is difficult to predict the impact of using a smaller network. On the one hand, the dataset is more uniform, with similar backgrounds for every image. On the other hand, all pollen grains look similar, so depth might be necessary to differentiate between them. In the first experiment, the model is trained with three different pre-trained feature extraction networks. The selected networks are given in Table 4.4

Table 4.4: Chosen feature extraction networks and relative size calculated by number of trainable parameters

Network	Relative size
ResNet 34	1.00
ResNet 18	0.34
MobileNet V2	0.02

4.5.2 Layer activation

The auxiliary structure also has opportunities for simplification. The primary rationale for SSD's multiple stacked feature maps is to allow for predictions at multiple scales in the image. However, pollen grains appear very uniform in size and shape throughout the dataset. The hypothesis is that some or most of the source feature maps can be removed from the model without any performance degradation. To test this, the model will be trained in multiple configurations where certain feature maps from the model are removed for each configuration. In practice, only the detection blocks fed by the feature maps are deactivated so that layers can be arbitrarily deactivated without disturbing successive layers of the model.

4.5.3 Minimum training sharpness

As observed, the dataset contains a wide variety of grains spanning many levels of blur. It remains unclear what impact blurry data has on the training of fully convolutional object detection systems. This experiment aims to explore if there is a correlation between the distribution of sharpness in the training data and model predictions. The model's performance is not of interest, but rather how the distribution of sharpness values in the training data changes the distribution of sharpness in the predictions. Sharpness can be measured both on the model's predictions and on the ground truths it does or does not detect.

For the experiment, the model is trained on various versions of the same dataset. Each model will be trained on a subset of the ground truth labels using a sharpness filter that prunes any ground truth below a certain threshold. Raising the threshold limits the model to only those pollen grains sharper than the threshold. Five thresholds are used, spanning a range from 0 to 0.08, which covers the lower half of sharpness values contained in the dataset.

4.5.4 Cross-sharpness inference

The final experiment aims to further explore what ability the model has to generalize outside of the sharpness values it is exposed to and whether or not a trained model can be transferred to different sharpness ranges. Similar to the previous experiment, the model is trained on a subset of ground truths, one with only the sharp ground truths and one with only the blurry ground truths. The same partition is also applied to the testing split. Together with a baseline containing all ground truths, three models and three test splits are created. Each of the three models will be tested on each of the three test partitions, creating a kind of sharpness confusion matrix.

Chapter 5

Results

This section will present and discuss the results from the experiments presented in Section 4.5. The modified SSD model presented in Section 4.3 is trained according to the training procedure given in Section 4.4. Section 5.1 presents the performance of the baseline model. The results and discussion of the experiments pertaining to RQ1 and RQ2 are presented in Section 5.2 and Section 5.3 respectively. Unless specified otherwise, all performance metrics given in this section are measured on the test split of the dataset. For measuring mAP, AP is computed according to Equation 2.1 with an acceptance threshold of 0.5 IoU.

5.1 Baseline model

The training procedure for the model is given in Figure 5.1 and exhibits a stochastic variation that is characteristic of mini-batch training. The confidence loss component accounts for most of the total loss throughout the training process. The components of the loss function (Equation 4.3) are independent, meaning their ratio does not indicate a difference in performance at the two different tasks. Regardless, the model does perform better at localization than classification. Logically this follows from the realization that pollen grains are all more easy to distinguish from the background than they are to classify. Therefore, the model can discriminate between background and pollen with relative ease but has a more challenging time classifying species.

The baseline model achieves a mAP of **95.2%** and an F_1 score of **86.8%**. Looking at predictions overall, the recall is very high at 99.1%, while precision is quite a bit lower at 77.2%. Figure 5.2 breaks down all the detections made by the model; the main observation is that false-positive predictions are almost the only source of error. 37% of false positives are spurious localizations, i.e., an unlabeled entity is identified as a pollen grain. These entities are either non-pollen particles or pollen grains from unlabeled species. The remaining 63% of false positives are misclassifications, i.e., the bounding box does overlap a ground truth, but the predicted class is incorrect.

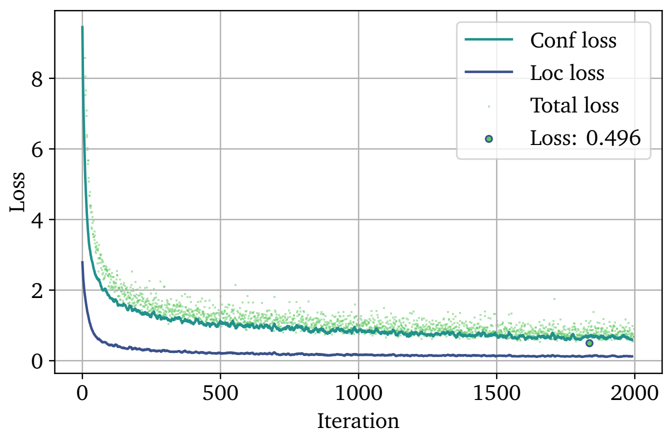


Figure 5.1: Training procedure for the baseline model over 2000 iterations. The iterations given on the horizontal axis represent one forward-backpropagation run with one mini-batch, while the mini-batch averaged loss is given in the vertical axis. The raw total loss values are shown with semi-transparent green points. The solid lines show the moving means for the two individual loss components. The green line shows the confidence loss component, while the blue line shows the localization loss component. The mini-batch with the lowest total loss is annotated with a green circle and occurs after the 1800th iteration with a loss of 0.496.

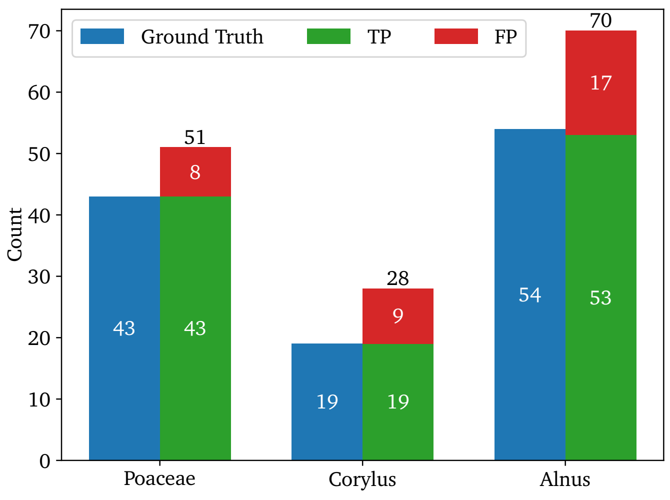


Figure 5.2: Breakdown of predictions by class. For each label, the first column gives the number of ground truth labels in the test split. The second column gives the number of predicted boxes for that class, broken down into true positive matches in green, and false negatives in red. The difference between ground truths and true positives is the number of false negatives.

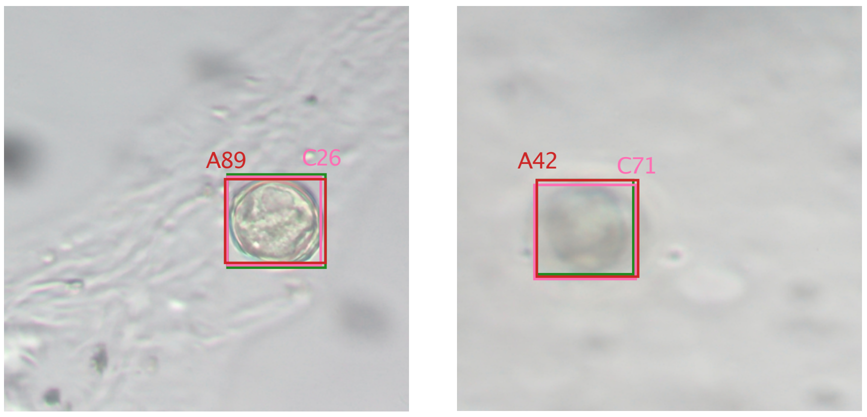


Figure 5.3: Two predictions with GT in green, TP in red, and FP in pink. The labels give the first letter of the class and the prediction confidence in the range [0,100]. In the left image, an FP box is predicted with lower confidence than the TP. In the right image, the FP Corylus prediction has higher confidence than the TP Alnus prediction.

Figure 5.3 shows two test samples containing overlapping true and false positive predictions. This is due to the NMS filtering algorithm being run for each class independently, which causes double predictions in cases where predictions for multiple classes are produced for the same pollen grain.

A remedy was implemented and tested where NMS was applied to all detections after the initial per-class NMS. The results from this were ambiguous; The precision did improve due to the decreased number of FP predictions, but the decrease in TP predictions caused a similar decrease in recall. In Figure 5.3 (right) for example, the correct prediction would get filtered out.

With the small number of classes in the dataset, the problem of double predictions is relatively small. However, a larger dataset could suffer more from this issue when there are more classes that all share characteristics. For the purposes of counting, it might not be preferable to prune these overlapping predictions but rather include and quantify the resulting overestimation as a part of the overall estimate.

5.2 Model simplification

The first two experiments aim to explore ways in which the computational complexity of the model can be reduced without sacrificing performance. In the first experiment, the feature extraction network is substituted with lightweight alternatives. In the second, the amount of default boxes is reduced by deactivating source feature maps.

5.2.1 Feature extractor

Table 5.1 gives the results from training the network with different feature extraction networks. As with the baseline, all the networks are designed for and trained on image classification datasets. Larger networks, such as the deeper ResNet versions, could not be tested due to the memory limitations of the GPU. There is a clear decrease in performance when using smaller networks, and we observe monotonically decreasing values for all performance metrics.

Interestingly, the recall remains almost unchanged, meaning that the model's ability to localize pollen grains is largely maintained. The error is in the form of noise being added on top of the correct predictions. The performance degradation is therefore primarily due to increased classification error, creating redundant, poor predictions. A possible explanation for this is that localization, in terms of feature requirements, only needs the model to learn simple features relating to the general shape and color of a pollen grain, while classification involves more complex features that the smaller networks are not able to capture.

This experiment helps reveal the apparent inequality between the two tasks that the model is learning, namely localization and classification. Localization seems to be a much easier task to learn, given how invariant the model is to the changing extractor while suffering a markedly higher degradation in classification performance.

Table 5.1: Summary of experimental results from testing various small feature extraction networks. The localization score refers to the share of detection that correctly localizes a GT, regardless of class. For each network, the number of trainable parameters for both the feature extractor and the whole model is given.

Network	Performance				Parameters	
	mAP	Precision	Recall	Loc.	Extractor	Total
ResNet 34	95.2	82.7	99.1	95.0	8.2×10^6	1.2×10^7
ResNet 18	92.6	81.9	97.4	92.8	2.8×10^6	6.7×10^6
MobileNet V2	90.4	67.9	98.3	86.3	1.3×10^5	9.6×10^5

5.2.2 Layer activation

In the next experiment, parts of the extra structure (shown as arrows in Figure 4.8) responsible for generating the detections at various scales are deactivated. The results are given in Table 5.2 and do not point towards any strong trend. The last three layers of the model are larger than almost all grains in the dataset, and even with cropping, it is likely that their predictions do not contribute to the loss because the default boxes they encode rarely, if ever, get matched with a ground truth. Their inclusion might result in parameters that do not get adequately trained, leading to those layers only producing noise instead of actual predictions.

For the models using only the first three layers, an improved mAP is observed, but the statistical significance is questionable. However, this does call into question the importance of the larger feature maps and strengthens the initial belief that the SSD model can be simplified without compromising performance.

Other object detection tasks may also have certain differences in the distribution of size and shape of objects between classes. The multiple layers then allow for specialization where some layers become better at detecting particular objects than others. With pollen, there again is a much more uniform dataset where size differences between classes are negligible, which might also explain why there does not seem to exist any strong relationship between layer activation and performance. This is compounded by the limited number of classes and could change if a larger, more diverse dataset was used.

Table 5.2: Summary of experimental results when deactivating various source layers. The source layers are ordered 1–6 with layer 1 being the most granular 38×38 feature map.

Source layer activation						Performance		
1	2	3	4	5	6	mAP	Precision	Recall
✓	✓	✓	✓	✓	✓	95.3	84.6	99.1
✓	✓	✓	✓	✓		95.5	77.4	99.1
✓	✓	✓	✓			95.3	81.2	98.7
✓	✓	✓				**96.4**	79.3	99.1
✓	✓					95.6	80.4	98.7
✓						96.3	83.6	99.1
	✓					95.7	85.5	99.1

5.3 Sharpness

The following two experiments look at how the sharpness of the ground truth data impacts model performance. In the first experiment, the blurry grains are filtered from the training set to see the effect this has on the inferences made by the model. In the second experiment, the dataset is split based on sharpness and measure the models' ability to make inferences across this sharpness boundary.

5.3.1 Minimum training sharpness

Our first experiment aims to explore how the sharpness of the training data affects the predictions that the model can make. The results indicate that the model struggles with making inferences on grains with lower sharpness when such grains are filtered out of the training data. Figure 5.4a breaks down the ground truth labels into true positives and false negatives, which reveals that the model is failing to predict pollen grains that have a sharpness below the minimum sharpness of the training data.

Looking at the sharpness of the predictions made by the model, there is a corollary trend in the false positive predictions. Figure 5.4b shows that restricting the training data creates what seems to be a threshold, below which the model does not make predictions. This indicates that the model loses the ability to localize blurry pollen grains, becoming blind to their existence. It seems clear that the features being learned from sharp images do not transfer to less sharp samples.

It is important to note that the sample size, especially in the case of false negatives, is relatively small, which weakens arguments made based on their distributions. The observed pattern could also be described as a type of ‘overfitting’ where the model will not generalize outside of the sharpness bound of the training data, but the implications this has for a hypothetical production system are unclear.

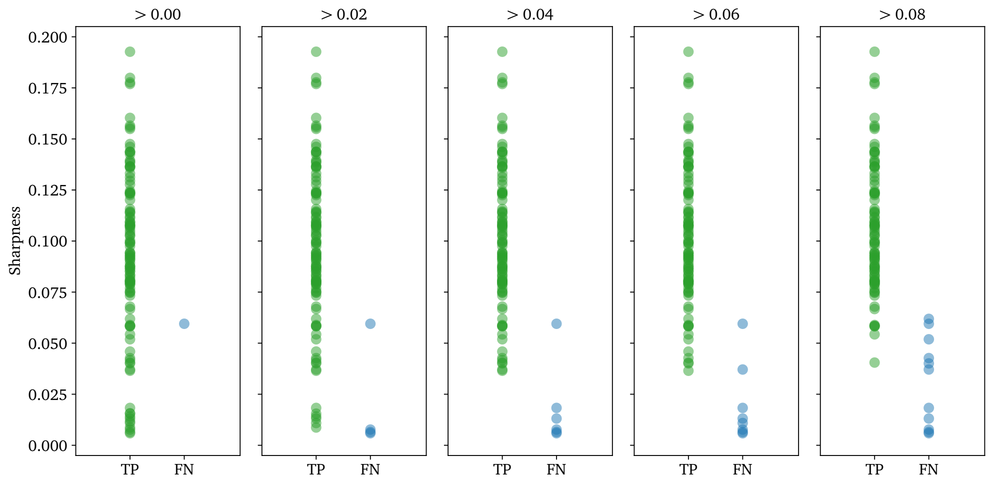
5.3.2 Cross-sharpness inference

The final experiment aims to explore how well the model can generalize to a sharpness range that differs entirely from its training data. The method expands upon the previous experiment by filtering both training and test data into sharp and blurry partitions. Figure 5.5 gives a summary of the experimental procedure. As might be expected, the model performs best in the scenarios where it is trained and tested on grains in the same range of sharpness, and it performs worse when there is no overlap between test sharpness and training sharpness.

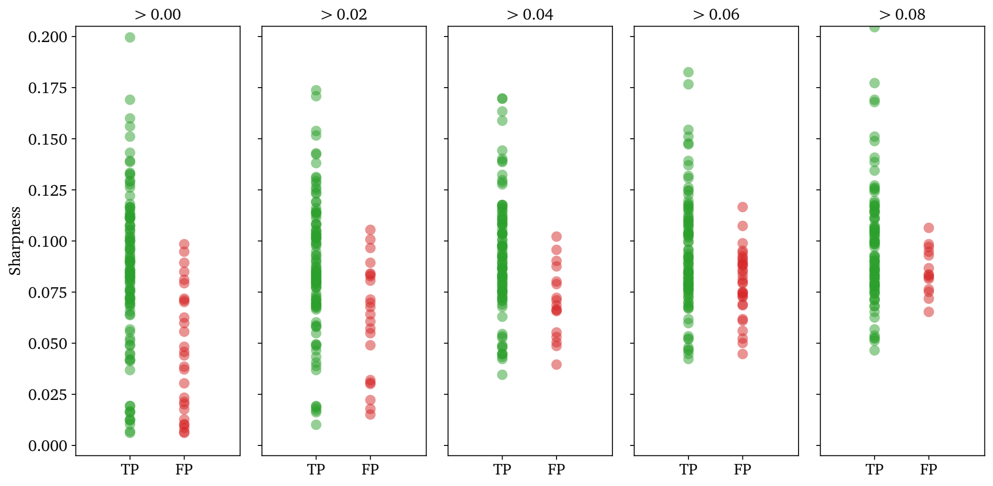
Interestingly, the model responds differently to being trained on only blurry data (the ‘blurry’ model) than being trained on only sharp data (the ‘sharp’ model). Seemingly, the blurry model outperforms the sharp model both when tested on the whole dataset and when tested across the sharpness boundary. It seems as though the same kind of overfitting observed in the previous experiment is not taking place for the blurry model, which is learning features from the blurry data that it can use to detect sharp pollen grains.

Figure 5.6 shows the precision and recall values separately and gives more insight into the differences between the two models. Looking at recall, the sharp model performs worse than the blurry model when tested across the sharpness boundary. Looking at precision, the opposite is true, however to a lesser degree. This shows that the models are underperforming for different reasons. The sharp model maintains its classification performance but fails to localize pollen grains that are too blurry. In contrast, the blurry model maintains its ability to localize all pollen grains but struggles with classifying them.

Similar to how simplifying the feature extractor leads to worse classification performance while localization is maintained, the same response is seen when the granular features are smoothed out by blur. What this implies for further development of this system is difficult to say and depends on the requirements of the system. In a scenario where the model only needs to work on sharp data, excluding blurry data could improve overall performance. However, one can easily



(a) Sharpness distribution of *ground truths* by prediction result in test split by minimum training sharpness



(b) Sharpness distribution of *model predictions* by prediction result in test split by minimum training sharpness

Figure 5.4: Five experimental runs are shown, ground truth labels are filtered out of the training set by the sharpness requirement given in the plot header. In a, sharpness is measured on original images using ground truth bounding boxes. In b, sharpness is measured on the resampled 300×300 pixel patches. This affects the scaling of the sharpness measure, and sharpness values from the two figures are therefore not directly comparable.

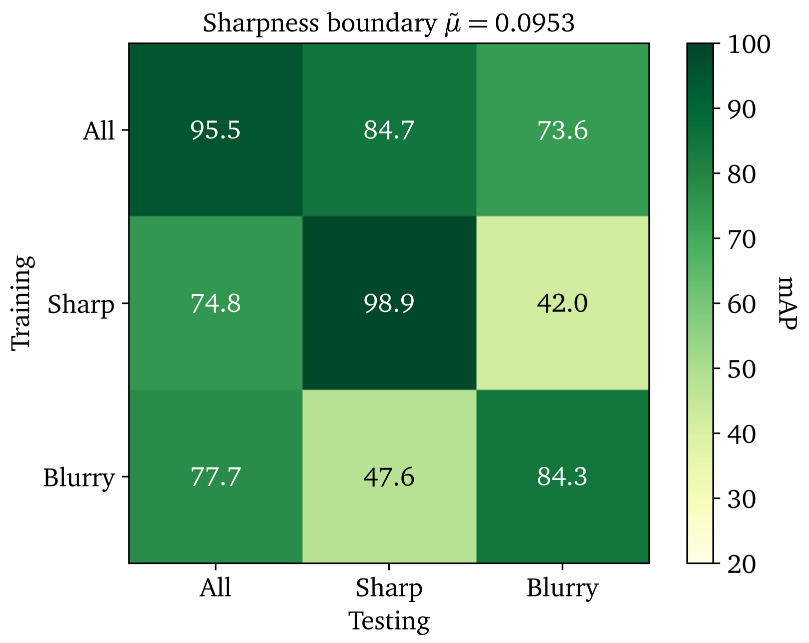


Figure 5.5: The confusion matrix shows the result from training and testing the model on three versions of the dataset. The median sharpness value, $\tilde{\mu}$, is used as a boundary and the dataset is divided into a sharp ($> \tilde{\mu}$) and a blurry ($< \tilde{\mu}$) data set. The same training/testing split is still used. The horizontal axis denotes the version of the dataset used when computing mAP, while the vertical axis denotes the dataset used when testing. Two quadrants stand out with significantly lower scores; these represent the scenarios where the model is trained on either the sharp or the blurry data and tested on the opposite.

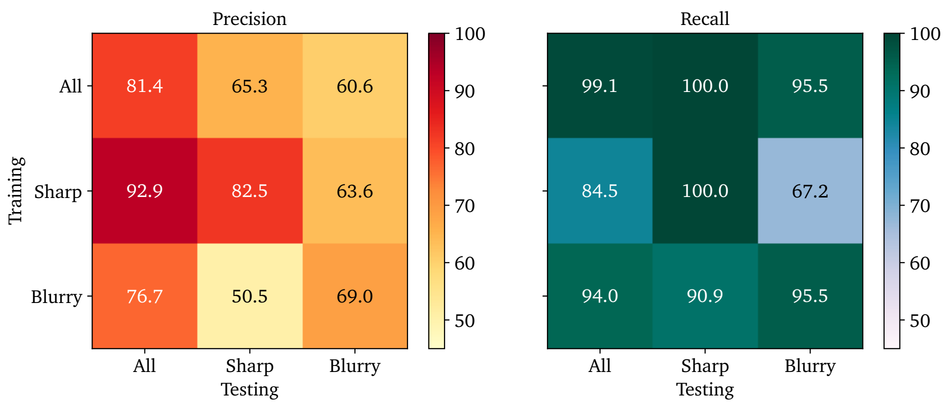


Figure 5.6: Breakdown of precision and recall values for cross-sharpness training. See Figure 5.5 for detailed explanation.

imagine a system that could benefit from being able to localize outside the current focal plane, e.g., in a search-based system that explores a slide live similar to the current manual method, the model could localize a blurry grain and refocus the microscope to classify it.

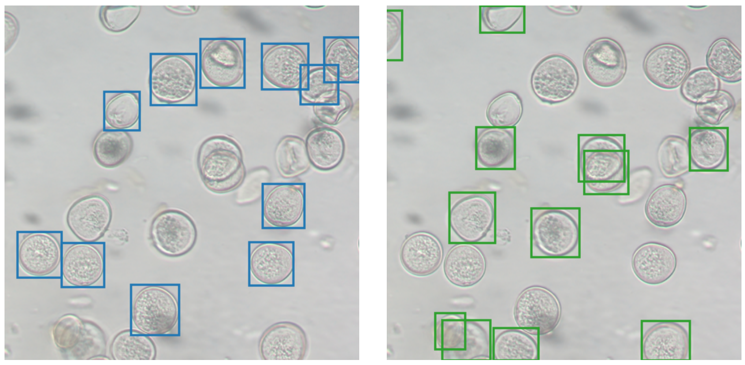

(a) Sharp labels **(b)** Blurry labels

Figure 5.7: Sample from the dataset. Ground truths in the sharp partition are drawn in blue, while the blurry are drawn in green.

While it seems to follow logically from the presence of more granular and detailed features in sharp data that the sharp model is more precise than its blurry counterpart, it is less obvious why it seemingly loses all ability to localize blurry grains. One possible explanation becomes apparent when looking not at the labeled data but at everything else the model is asked to ignore.

As a result of the data collection procedure, where at least one pollen grain is entirely in focus in every sample, the sharpest object in any given sample is always a pollen grain. Figure 5.7 shows an example of the split, and reveals that all in-focus objects are labeled ground truths. This trend holds over the entire dataset, where almost every object that appears in focus is a labeled ground truth. The inverse assumption that everything out of focus is not a ground truth does not hold, except in the sharp training split. So while the sharp model can rely almost entirely on the presence of sharp features when learning ‘what is a pollen grain?’, the blurry and baseline models must also learn to differentiate between out of focus pollen grains and out of focus background.

The boundary value is set to the median sharpness value over the total training split. This was done to give each model the same number of ground truths. While this is a valid decision, looking at Figure 5.7, many pollen grains in the blurry category seem pretty sharp to the naked eye. With a larger dataset, it would be preferable to run this experiment with multiple splits. The optimal sharpness distribution remains unknown, although evidence has been presented to support the claim that excluding blurry data causes the model to fixate on only the sharp features in images.

5.4 Summary

The final model performs well at both localizing and classifying pollen grains but tends to return multiple bounding boxes for some pollen grains. However, as part of a counting system, this might not be a significant issue. The model is able to identify almost every grain in the test set, which is promising for further development.

The experiments reveal that while identifying pollen grains is quite an easy task, discriminating between species is considerably more difficult. The model's classification performance is highly dependent on the quality of the features produced by the feature extractor.

Most interesting is the impact training data sharpness has on the model. Excluding un-sharp data from the model's training data causes the model to fixate on sharpness, reducing the model's ability to identify grains that appear less sharp. Including blurry samples seemingly forces the model to learn a broader range of features.

Chapter 6

Conclusion

In regards to the two initial research questions posed in this thesis and based upon the results of the experiments presented in Chapter 5, the following is given as the main conclusions of this work,

1. For the task of differentiating pollen grains, reducing the size of the network negatively affects precision. However, the uniformity of the objects in the dataset allows for a reduction in the number of detections made by the model without any apparent performance effects.
2. Excluding multifocal data from training causes a fixation on sharp features, inhibiting the model from localizing pollen grains only slightly outside the focal range of the training data.
3. Excluding multifocal data creates a more precise model with lower recall than a model trained on multifocal data.

The solution presented is a fully convolutional deep neural network capable of locating and classifying pollen grains from microscopic imaging data from a standard stereoscopic microscope. The performance of the presented model supports the more general claim that locating pollen is well suited for a CNN based solution. This first attempt using a CNN model shows that both the task of localizing and classifying pollen grains is solvable with a fully convolutional model. An adapted SSD model was used for this project, but future work may find that other single shot models, such as YOLO could provide better results. The trained model is by itself not a major contribution of this work but a proof of concept of the general approach.

The limits of this method are unknown, but the results presented indicate that localization is a far simpler task than classification, which could be the limiting factor in the scaling of this model to a wider set of classes.

This work makes two main contributions to the joint fields of palynology and machine learning:

1. A new, relatively large, pollen detection dataset totaling 701 sample images

and 6384 ground truth labels of Norwegian air-born pollen. As of writing, a dataset with both localizations and class labels has not been presented in the literature.

2. Evidence showing a benefit to using un-sharp data in the training of convolutional object detection models in domains using microscopic imaging data.

6.1 Future Work

The most important contribution is in reference to the use of unfocused data when training detection models, which to the knowledge of the author, is a novel discovery. Based on this and the weaknesses of this thesis, multiple paths can be explored in future work, which builds on this work.

Extending the dataset

The size of the dataset could pose a threat to the external validity of the model's performance. The results show that the model performs better at localization than at classification in a dataset containing three distinct classes. Within a domain where most potential classes share similar features, it is unknown how the model would perform with a dataset containing more classes and if it could maintain its ability to label classes correctly. Therefore, it cannot be concluded that the model would perform at a similar level of precision in datasets with more classes. Extending the number of classes in the dataset with more samples is the only way to verify this.

A different way of extending the existing dataset is by using synthesized data, which could significantly improve performance. New sample images could be artificially generated by rearranging ground truths and moving them in the focal plane.

Multifocal input

The results assume a link between perceived sharpness and focal planes in the sample images without a direct correlation being shown. This perceived sharpness is used to validate the sharpness measure, which in turn is used to analyze the model in relation to data sharpness. To an extent, this assumption does hold; when the focal plane lies above or below a pollen grain, perceived sharpness changes with the focal plane. However, on a more granular level, how the sharpness measure corresponds to the location of the focal plane when it lies within a pollen grain is less clear. From observation, maximum sharpness does seem to occur at the center of the pollen grain when the diameter of the grain is largest.

Using stacks of images from multiple focal planes as the input to the model, instead of a single image, could enhance performance by providing the model a detailed view of the entire surface of each pollen grain.

Counting algorithm

Turning the model's raw output into an accurate pollen grain count for an entire microscopic slide is a non-trivial task. Pollen grains may be placed directly over one another, only distinguishable by running the model over all focal planes. One possible approach is to use video streams from a microscope, taken while it slides its focal plane across a microscope slide and, from all the individual detections and sharpness values, create a 3D positional model of every detected grain. This would enable counting in three dimensions and could require the creation of a novel filtering algorithm that extends the concept of overlapping predictions into three dimensions.

Live counting

A second possibility for a functioning counting system is to embed the model in a live system that controls the motion of a microscope along all three axes, similar to the current manual method with human operators. The model's output could be used to guide a new search algorithm that moves over a slide, building a complete detection model for the entire slide in all three axes.

Bibliography

Askin, R. and S. Jacobson (2003). “Palynology”. In: *Encyclopedia of Physical Science and Technology*. Journal Abbreviation: Encyclopedia of Physical Science and Technology, pp. 563–578. DOI: 10.1016/B0-12-227410-5/00930-3.

Bodla, Navaneeth et al. (2017). *Soft-NMS – Improving Object Detection With One Line of Code*. arXiv: 1704.04503 [cs.CV].

Boser, Bernhard E, Isabelle M Guyon, and Vladimir N Vapnik (n.d.). “A training algorithm for optimal margin classifiers”. In: *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pp. 144–152.

Chibuta, S. and A. C. Acar (2020). “Real-time Malaria Parasite Screening in Thick Blood Smears for Low-Resource Setting”. In: *Journal of Digital Imaging* 33.3, pp. 763–775. DOI: 10.1007/s10278-019-00284-2.

Daood, A., E. Ribeiro, and M. Bush (2016). “Pollen grain recognition using deep learning”. en. In: *Advances in Visual Computing*. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 321–330. DOI: 10.1007/978-3-319-50835-1_30.

Daood, Amar, Eraldo Ribeiro, and Mark Bush (2016). “Pollen Recognition Using a Multi-Layer Hierarchical Classifier”. In: *IEEE International Conference on Pattern Recognition (ICPR)*. DOI: 10.1007/978-3-319-50835-1_30.

— (2018). “Sequential Recognition of Pollen Grain Z-Stacks by Combining CNN and RNN”. In: *Proceedings of the International Florida Artificial Intelligence Research Society Conference, FLAIRS, Melbourne, Florida, USA. May 21–23. Published by The AAAI Press, Palo Alto, California.*

De, Kanjar and V. Masilamani (2013). “Image Sharpness Measure for Blurred Images in Frequency Domain”. In: *Procedia Engineering* 64. International Conference on Design and Manufacturing (IConDM2013), pp. 149–158. DOI: 10.1016/j.proeng.2013.09.086.

Everingham, Mark et al. (2010). “The pascal visual object classes (voc) challenge”. In: *International journal of computer vision* 88.2, pp. 303–338. DOI: 10.1007/s11263-009-0275-4.

France, I et al. (2000). “A new approach to automated pollen analysis”. en. In: *Quaternary Science Reviews* 19, pp. 537–546. DOI: 10.1016/S0277-3791(99)00021-9.

Gallardo-Caballero, Ramón et al. (2019). “Precise pollen grain detection in bright field microscopy using deep learning techniques”. In: *Sensors (Basel, Switzerland)* 19. DOI: 10.3390/s19163583.

Girshick, Ross et al. (2014). “Rich feature hierarchies for accurate object detection and semantic segmentation”. In: arXiv: 1311.2524 [cs.CV].

Halbritter, Heidemarie et al. (2018). “Methods in Palynology”. In: *Illustrated Pollen Terminology*. Springer International Publishing, pp. 97–127. DOI: 10.1007/978-3-319-71365-6_6.

He, Kaiming et al. (2015). “Deep Residual Learning for Image Recognition”. In: arXiv: 1512.03385 [cs.CV].

Hernández-García, Alex and Peter König (2019). “Further Advantages of Data Augmentation on Convolutional Neural Networks”. In: *CoRR* abs/1906.11052. arXiv: 1906.11052.

Ioffe, Sergey and Christian Szegedy (2015). “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: arXiv: 1502.03167 [cs.LG].

Iqbal, Haris (2018). *HarisIqbal88/PlotNeuralNet*. Version v1.0.0. DOI: 10.5281/zenodo.2526396.

Islam, M. and M. Alam (2019). “A Machine Learning Approach of Automatic Identification and Counting of Blood Cells”. In: *Healthcare Technology Letters* 6. DOI: 10.1049/htl.2018.5098.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton (2017). “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Commun. ACM* 60.6, pp. 84–90. DOI: 10.1145/3065386.

Langford, M., G. E. Taylor, and J. R. Flenley (1990). “Computerized identification of pollen grains by texture analysis”. en. In: *Review of Palaeobotany and Palynology*. The Proceedings of the 7th International Palynological Congress (Part I) 64, pp. 197–203. DOI: 10.1016/0034-6667(90)90133-4.

LeCun, Yann (1989). “Generalization and network design strategies”. In: *Connectionism in perspective*. Ed. by R. Pfeifer et al. Elsevier.

LeCun, Yann, Bernhard Boser, et al. (1990). “Handwritten digit recognition with a back-propagation network”. In: *Advances in neural information processing systems* 2, pp. 396–404.

LeCun, Yann, Léon Bottou, et al. (1998). “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11, pp. 2278–2324. DOI: 10.1109/5.726791.

Li, Ping and J. R. Flenley (1999). “Pollen texture identification using neural networks”. In: *Grana* 38, pp. 59–64. DOI: 10.1080/001731300750044717.

Lin, Min, Qiang Chen, and Shuicheng Yan (2014). “Network In Network”. In: arXiv: 1312.4400 [cs.NE].

Liu, W., L. Cheng, and D. Meng (2018). “Brain slices microscopic detection using simplified SSD with Cycle-GAN data augmentation”. In: *Neural Information Processing*. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 454–463. DOI: 10.1007/978-3-030-04212-7_40.

Liu, Wei et al. (2016). “SSD: Single Shot MultiBox Detector”. In: *Lecture Notes in Computer Science*, pp. 21–37. DOI: 10.1007/978-3-319-46448-0_2.

El-Melegy, M., D. Mohamed, and T. El-Melegy (2019). “Automatic detection of tuberculosis bacilli from microscopic sputum smear mmages using faster R-CNN, transfer learning and augmentation”. In: *Pattern Recognition and Image*

Analysis. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 270–278. DOI: 10.1007/978-3-030-31332-6_24.

Pan, Sinno Jialin and Qiang Yang (2010). "A Survey on Transfer Learning". In: *IEEE Transactions on Knowledge and Data Engineering* 22.10, pp. 1345–1359. DOI: 10.1109/TKDE.2009.191.

Paszke, Adam et al. (2019). "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: arXiv: 1912.01703 [cs.LG].

Redmon, Joseph et al. (2016). "You Only Look Once: Unified, Real-Time Object Detection". In: arXiv: 1506.02640 [cs.CV].

Sevillano, V. and J. L. Aznarte (2018). "Improving classification of pollen grain images of the POLEN23E dataset through three different applications of deep learning convolutional neural networks". en. In: *PLOS ONE* 13. DOI: 10.1371/journal.pone.0201807.

Simonyan, Karen and Andrew Zisserman (2015). "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: arXiv: 1409.1556 [cs.CV].

Szegedy, Christian et al. (2014). "Going Deeper with Convolutions". In: arXiv: 1409.4842 [cs.CV].

NTNU
Norwegian University of
Science and Technology