Universiteit Leiden

Master Computer Science

One-Stage Neural Network Analysis
for allergenic tree Pollen Counting

Name:	Casper Stiekema
Student ID:	s2584832
Date:	29/08/2025
Specialisation:	Artificial Intelligence
1st supervisor:	Lu Cao
2nd supervisor:	Fons Verbeek

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

keywords

Pollen counting, one-stage models, deep convolutional neural networks, YOLO, RetinaNet, DINO, automated image detection

Abstract

Allergic rhitinis caused by exposure to pollen is one of the most common allergies in the the world. As a result, pollen monitoring becomes an increasingly important method for research on hypoallergenic designs of urban green. Unfortunately, manual pollen counting is slow and labour-intensive and as such large scale and real-time research is impossible. However, with the advancement of visual deep learning models, an opportunity is presented to detect and classify the pollen automatically. In this field, one-stage models combine detection and classification within a single architecture. They have steadily improved in precision and robustness, while also offering lower training and inference costs compared to two-stage models. As such, this study aims to evaluate benchmark one-stage models, YOLO, RetinaNet and DINO(DETR-Swin), for the detection and classification of three common allergenic urban tree pollen species in the Netherlands, Betula pendula, Callitropsis nootkatensis and Chamaecyparis lawsoniana. Doing so, we aim to create a system which offers a scalable alternative to manual pollen counting.

1 Introduction

Allergic rhitinis, or hayfever, is increasingly common, with the number of affected people increasing for both children and adults [Sheikh et al., 2009]. This rise is exacerbated by environmental changes like urbanisation and climate dynamics, which does not look to alter its course any time soon. Consequently, as these allergies become an increasingly significant factor for public health, so does our interest in the airborne pollen causing them. The timing and amount of pollen released into the air is difficult to estimate, being influenced by a myriad of factors like local temperatures, humidity, wind and air pollution. Moreover, the allergenicity of airborne pollen can be further affected by the presence of other pollen species and overall air quality [Gisler, 2021]. As a result, fast and accurate monitoring of airborne pollen is vital for both academic study and economics-driven research. It could provide an early warning system, as well as providing the means to create an environment that keeps highly allergenic pollen further from high-risk areas (such as schools or hospitals). Yet the path to achieve this still requires work. Current airborne pollen counting relies on manual microscopy to identify pollen in samples, a process that is both time-consuming and demanding of expert knowledge. Not only this, but even with expert knowledge miscounts can occur, as counting customs can differ from person to person, and in some common urban taxa, only a genus- or family-level identification is possible with the human eye [Buters et al., 2024]. Advancements in computer vision and deep learning, particularly in Convolutional Neural Networks (CNNs), offer promising prospects for automating this process. Existing CNNs have been used on single pollen grain images, using a model like ResNet [Li et al., 2023, Rostami et al., 2023] to classify pollen into the right species. These can be used together with detection methods such as image processing, machine learning, or other deep learning models, to be able to count pollen on large scans like those from pollensniffers [del Pozo-Banos et al., 2015]. Such methods, in which recognition and classification are split into different tasks, are referred to as two-stage models. In this paper we aim to achieve both the identification (finding the pollen in the image) and classification in a single model. Such one-stage models have already been created to classify a limited number of pollen grains in other countries [Zhang et al., 2024]. One-stage models can give the advantages like requiring less data and inference and could accordingly be easier to expand to detect and classify other pollen. Furthermore, they should be more effective at counting pollen in varied environments compared to rule-based classical processing. Since pollensniffer slides can contain a large assortment of miscellaneous particles, this can influence to the accuracy of the model counting.

Therefore, our focus is on counting airborne allergenic tree pollen that are common in urban environments in the Netherlands. These allergenic pollen have the most contact with humans, and exhibit cross-allergenic behaviour. Therefore, the scientific impact of classifying them is of greater impact than other allergenic tree pollen. For our study, the processed dataset of microscopy samples contain the following species: Betula pendula, Callitropsis nootkatensis and Chamaecyparis lawsoniana. Although this is a small representative of all common tree pollen in the Netherlands, the focus of the study is to find and optimise models, which can be expanded to include many more once operational.

The goal of this study is to evaluate the effectiveness of modern one-stage object detection models in automating the identification and classification of tree pollen in microscope imagery, offering a faster and scalable alternative to manual counting. To achieve this, we develop a comprehensive pipeline that includes preprocessing using multiple projection methods, exploration of classical image segmentation for annotation, and model selection, training and parameter-optimisation. Three structurally distinct one-stage detectors are selected as representative benchmarks, YOLOv8, RetinaNet, and DINO (DETR-Swin). The following chapter will provide more context into the research problem, covering public health relevance of pollen monitoring, previous applications of deep learning for pollen classification, and an overview of the selected one-stage object detection models. The next section on methodology will outline the projects system to achieve this objective, from data preprocessing to tested annotation methods and model training, hyper-optimisation, and performance evaluation. In our results, we show the outcomes of our experiments, comparing different projection types, the three model architectures, and model hyper-optimisation. These findings and limitations are further analysed in our discussion, and potential regions of interest are then covered in the future works section. Finally, the conclusion reflects on the work and potential for integrating such models into real-world pollen monitoring workflows.

2 Background

2.1 The Impact of Pollen in Public Health

Pollen plays a significant role in many areas of human life, prompting research across fields like agriculture, ecology, and health. Nevertheless, the presence of pollen in our environment would attract little attention were it not for the large number of people that are allergic to it. Pollen allergies affect an increasing portion of the global population, with over 40% of Europeans now estimated to suffer from some form of allergic rhinitis, commonly known as hay fever [D'Amato et al., 2017] . Recent studies show not only a rise in the number of individuals affected, but also in the intensity of allergic responses per person. This is most likely due to both environmental changes and increased urban exposure [Gisler, 2021]. In densely populated environments, individuals are more frequently exposed to allergenic urban tree pollen such as *Betula pendula*, *Callitropsis nootkatensis* and *Chamaecyparis lawsoniana*, which dominate city landscapes in the Netherlands due to their popularity in urban planning and as ornamentals in private gardens. Not just the number of pollen, but the species and combinations of airborne pollen can have a large effect on the allergic reactions. In particular, the latter two species are notable for their cross-allergenic effects, as well as similar morphology.
Understanding pollen and its role in allergic disease is of considerable social and economic importance, as hay fever significantly impacts quality of life, affecting sleep, concentration, and physical performance. In Europe alone, the annual economic burden of allergic diseases is estimated between €55 and €151 billion per year, largely due to lost productivity and healthcare costs [Zuberbier et al., 2014]. Of all allergies, hayfever is by far the most common. To make effective change, local, high-accuracy pollen monitoring data is vital for developing accurate allergen forecasts, especially during peak seasons. These forecasts play an increasingly important role in informing urban planning, health policies, and individual behaviour, such as adjusting medication or planning outdoor activities.

2.2 Current Monitoring Methods

The Pollensniffer is one of the most successful methods for airborne pollen monitoring, being both portable and approximately 5–6 times more efficient in collecting pollen compared to other static methods [de Weger et al., 2020]. Similar to classical Hirst pollen traps, it uses a vacuum mechanism that draws airborne particles onto adhesive-coated Melinex strip, coated with petroleum jelly (Vaseline) to catch pollen. These slides are then manually analysed using brightfield microscopy, often combined with staining to enhance contrast and visibility of pollen features. Trained scientists then review each slide under high magnification, usually between 100- and 400-times magnification, identifying and counting pollen grains manually. This process is both labour-intensive and time-consuming, with a single slide requiring several hours of review, especially when pollen densities and diversity are high [LeCun et al., 2015]. Moreover, taxonomic identification is often limited to the genus or family level due to overlapping morphological features. This is especially challenging for many conifer species, which are abundant in urban environments but difficult to distinguish microscopically. As a result, traditional methods often lack the precision required for species-level forecasting or fine-scale environmental studies.
Automation of this process could greatly benefit the study of pollen. To start off, labelling takes valuable time from experts. Automation can also solve the problem of certain inter species level classification, finding features the human eye might not. It could aid in situation where samples are compromised even for experts, such as through debris, air bubbles or vaseline. These issues compound when larger scale studies are implemented, such as monitoring in an area over a time period or getting significant results in areas where pollen are sparser. Automation can come into play by providing a fast, stable, and reliable method of measuring, which can catch patterns the human eye cannot.

2.3 Deep Learning for Pollen Classification

There are many ways of applying machine learning to complete the pollen counting task. Although many methods are possible, like support vector machines, the most prominent in the current field of AI application is deep learning [del Pozo-Banos et al., 2015]. Unlike other forms of machine learning, deep learning does not rely on handcrafted features to identify objects. Instead it trains weights in hidden layers to self determine which combination of inputs is important for classification or clustering. This is achieved through the backpropagation algorithm, which iteratively adjusts a layer's weights by minimizing a loss function through gradient descent [LeCun et al., 2015]. The structure of how data is fed and altered through a layer or model is called the model or layer architecture. In the case of deep learning for visual tasks, the form of architecture is usually

done with a Convolutional Neural Network, or CNN. CNNs are a well-known architecture, heavily inspired by the perception mechanisms of living organisms [Khan et al., 2020]. Unlike standard neural networks, which assign weights to specific pixel combinations, CNNs introduce convolutional layers that apply a shared kernel across the entire input image. This approach significantly reduces the number of parameters, allowing for deeper networks without impairing backpropagation. The early layers in a CNN are particularly effective at capturing location-invariant features such as edges and corners. As the network depth increases, CNNs can learn to recognize higher-level abstractions, such as distinguishing between faces or telling the time from a clock. This automated feature extraction is especially advantageous in domains like computer vision, where manually engineering features is difficult. It also enhances generalization and reduces the need for extensive preprocessing.
As CNNs evolved, several enhancements were introduced to further enhance the processing. These include pooling layers, which reduce the spatial dimensions and the number of connections between layers [Khan et al., 2020]; improved activation functions such as the Leaky Rectified Linear Unit (Leaky ReLU), which helps reduce the "dying ReLU" problem [Maas et al., 2013]; and various improvements to loss functions, regularization techniques, and weight decay [Khan et al., 2020]. Overall, CNNs have proven to be highly effective and well-optimized for end-to-end supervised computer vision tasks, often requiring little to no preprocessing [Krizhevsky et al., 2012].

2.4 Limitations of Existing CNN Pollen Studies

Visual deep learning models have been put to use for pollen classification already. One prominent study by Li applied CNNs to pollen classification using a dataset of single-pollen grain images [Li et al., 2023]. They trained ResNet-based architectures to distinguish among different pollen species, achieving high accuracy on curated and pre-segmented samples. Similarly, Rostami et al. 2023, introduced the "Efficient Pollen Grain Classification" approach, applying lightweight CNN architectures to improve computational efficiency without compromising accuracy [Rostami et al., 2023]. This model reached a very high accuracy, but these approaches assumed ideal conditions: uniform image backgrounds, centered single-particle images, and no environmental noise or occlusion. DeepPollenCount, another recent project, takes a similar approach using pre-segmented pollen grain images and CNN classifiers such as ResNet and MobileNet [Zhang et al., 2024]. While DeepPollenCount does consider detection counts in its name, in practice it still relies on isolated pollen grains with consistent backgrounds, rather than detecting multiple grains in complex environments. Most previous approaches optimize classification but skip the object detection step, assuming the pollen grain has already been segmented.
This fixed-image classification setup diverges significantly from real-world applications. Pollensniffer slides and similar sources feature visual clutter: overlapping pollen grains, air bubbles, debris, and a broad diversity of non-pollen particles. The lack of end-to-end detection-and-classification models means that current methods are poorly suited for general deployment without extensive preprocessing or manual slide segmentation. Despite these strengths, most deep learning methods in pollen research have not incorporated detection pipelines or object-localization models. Addressing this gap by integrating detection and classification into a unified pipeline may improve the applicability of these models to real-world tasks such as environmental monitoring or public health alerts.

2.5 Object Detection Models: One-Stage vs. Two-Stage

While image classification assigns a label to an entire image, object detection tasks are more complex: they involve both localizing regions which contain objects and assigning class labels to each localized region. This distinction can be especially challenging in tasks like pollen detection, where objects can be stacked, partially occluded by other particles, or distorted by being out of focus or in an air bubble and the number of grains in a single field of view can vary wildly.
Two-stage models, such as Faster R-CNN [Ren et al., 2015], separate the detection and classification steps into multiple modules. The first generates region proposals and the second classifies and refines each proposal, leading to high detection accuracy but with a cost in computational complexity and inference time. One-stage detectors, by contrast, such as YOLO [Redmon et al., 2016] and RetinaNet [Lin et al., 2017], alter this process by predicting bounding boxes and class labels in a single forward pass. They do this by connecting the feature detection model, or backbone, and the detection and classification model, or head, at multiple feature levels. Often, the model backbone uses anchor boxes to denote regions of interest which the heads classify. This results in faster inference times, making them well-suited for real-time or high-throughput applications.

One reason for this is that one-stage models avoid the additional region proposal stage present in two-stage architectures. Since the one-stage model heads build upon feature signals at multiple levels from the backbone, this means predictions can re-use shared feature maps, reducing redundant computation. Furthermore, since more candidate regions (anchors or feature points) are evaluated per image this results in more feedback per image trained. Although traditionally seen as slightly less accurate than two-stage models, one-stage detectors have steadily improved in precision and robustness; in some cases surpassing two-stage models on lower training data.
The relevance of one-stage models has steadily grown in fields like microscopy, where large numbers of small, similar objects must be located and identified. Studies have successfully applied YOLO and related architectures for detecting cells, parasites eggs, and other biological entities in dense, cluttered visual environments [Wu et al., 2025, Kumar et al., 2023]. These use cases closely mirror the challenges posed by airborne pollen detection on microscope slides, where efficient multi-object recognition in noisy backgrounds is critical.

2.6 Selected One-Stage Models: YOLO, RetinaNet, DETR-Swin

In this study, we selected three state-of-the-art one-stage object detectors, YOLO, RetinaNet, and DETR-Swin, based on their distinct architectural strengths and their proven performance in other studies [Wu et al., 2025, Tian et al., 2024, Liao et al., 2024]. By using models with clear differences in structure they can serve as benchmarks when attempting to solve similar image analysis tasks. YOLOv8 is a fast, anchor-free detector with a decoupled head architecture optimized for real-time applications. RetinaNet, equipped with a ResNet-50 backbone, employs a feature pyramid network (FPN) to enable multi-scale detection and uses dense anchors to localize small and variably sized objects, making it particularly suited for high-resolution slides where pollen grains vary in scale. Finally, DINO (based on the DETR family) incorporates transformer-based attention mechanisms with a Swin Transformer backbone, allowing it to handle complex spatial relationships like clusters.

2.6.1 YOLO (You Only Look Once)

YOLOv8 is included in this study as a representative one-stage object detection model that adopts an anchor-free design, optimised to be able to process even live video. You Only Look Once (YOLO) models have evolved across multiple iterations, with YOLOv8 introducing a decoupled head and refined detection strategies compared to earlier versions [Jocher et al., 2023]. YOLOv8 predicts object centres and box dimensions directly, relying on point-based detection, being able to process even live feed videos. This approach removes the need for an anchor generation step, where fixed candidate boxes, or anchors, of varying sizes and aspect ratios are created.
The YOLOv8 architecture consists of a convolutional backbone, a feature aggregation network, and a detection head. The backbone is designed to encode input images into multi-scale feature maps, which are then combined in the “neck” of the network. YOLOv8 uses a PANet-style (Path Aggregation) structure, combining top-down and bottom-up connections between feature maps. The detection head operates on these combined feature maps to simultaneously predict object classes and their corresponding box coordinates. It does this by dividing the image into smaller squares, which are populated with bounding boxes and assigning with an "objectness" score, whether it is an object or background. These boxes are then assigned a class with a confidence metric. From these metrics, Non-Maximum Suppression(NMS) picks the most plausible boxes and suppresses the rest [Endo et al., 2024].
YOLOv8 applies a decoupled design for classification and regression branches, allowing the model to optimize these objectives separately. In the context of pollen grain detection, YOLO offers several advantages. It performs well on medium-scale objects and is computationally lightweight. It may struggle with small objects or highly imbalanced classes, but these limitations can be mitigated using tiling so that images do not need to be downscaled and balanced data preparation. For this task, if YOLO performs to a decent accuracy, it could prove fruitful to develop, since it will take less time to add and train new classes of pollen.

2.6.2 RetinaNet

RetinaNet is included in this study as a representative one-stage object detector based on convolutional backbones and anchor box mechanisms. It combines a ResNet-50 backbone with a Feature Pyramid Network (FPN) to extract multi-scale feature maps from an input image [Lin et al., 2017]. The use of an FPN allows RetinaNet to detect objects at different spatial resolutions by creating hierarchical feature maps with semantically strong representations at each level.

At each level of the pyramid, RetinaNet applies two subnetworks: one for classifying object categories and one for regressing bounding box coordinates. The detection process relies on predefined anchor boxes of multiple aspect ratios and scales, densely applied across each feature map. This structure enables the model to handle variation in object size and shape within an image, depending on the training configuration. A key contribution of RetinaNet is the introduction of the Focal Loss function. In standard object detection tasks, a common challenge is the overwhelming number of background (negative) anchor boxes compared to foreground (positive) ones. Focal Loss reduces the loss contribution from well-classified examples, allowing the model to focus learning on harder, misclassified ones. This property is useful in contexts where object instances are unevenly distributed or when hard-to-distinguish classes are present or when visually similar classes must be distinguished, making it a strong candidate for pollen detection tasks [Lin et al., 2017]. RetinaNet offers a balance between inference efficiency and detection accuracy, without requiring a region proposal stage as in two-stage detectors. It remains widely adopted in computer vision research and serves as a well-documented benchmark for one-stage detection tasks with convolutional feature extractors.

2.6.3 DINO DETR-Swin

DINO is the model chosen to represent the one-stage transformer models. It is a form of DETR-Swin; combining a Swin Transformer backbone with a Detection Transformer (DETR) decoder. DETR introduced end-to-end object detection via sequence-to-sequence prediction [Carion et al., 2020]. This idea came originally for the development of natural language models, however they were found to be rather successful at other tasks, like object detection [Vaswani et al., 2017].

Our model employs a Swin Transformer as its backbone. This architecture partitions the input image into fixed-size, non-overlapping windows and computes multi-head self-attention (MHSA) within each window. To enable global context learning, Swin uses a shifted window mechanism, offsetting windows between layers. This design captures both fine-grained local features and long-range spatial dependencies while preserving linear computational complexity [Liu et al., 2021].

DINO improves upon earlier DETR-style models by addressing common limitations such as slow convergence and difficulty detecting small objects. It does this by adding noisy labels and locations during training such that the model should converge faster and increase in robustness. It uses auxiliary loss, which means that it evaluates hidden decoder layer results as well as the final layer, to increase stability. Furthermore, DINO enhances Hungarian matching, a one-to-one assignment algorithm that aligns predicted and ground truth objects without relying on non-maximum suppression. This set-based prediction paradigm avoids redundancy and is particularly well-suited to tasks with many small, overlapping instances, such as with images of slides of stacked pollen. These improvements could prove crucial in distinguishing visually similar species of pollen grains.

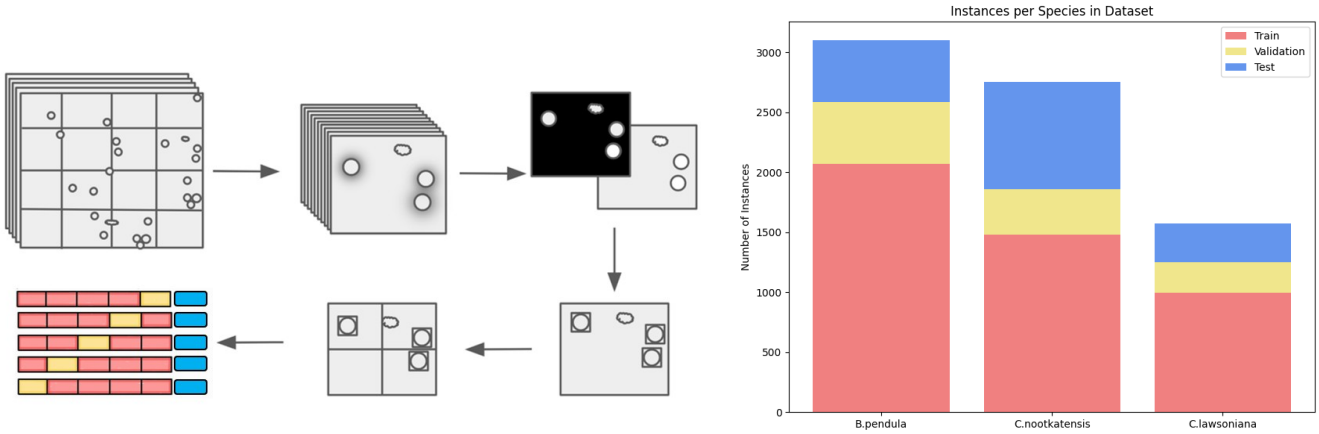


Figure 1: Left: Visualisations of our pre-processing pipeline. This includes projecting the z-stack, labelling, tiling, formatting and k-fold splitting of the train and validation datasets. Right: Distribution of instances per species and dataset split.

3 Method

In the following parts we will briefly explain the methodology for the project. For clarity, a flowchart has been provided in Figure 1 to visualise the processing steps. To summarise, this process starts with a complete catkin or pollensniffer scan, which is split into many sub-images of different locations and visual depths. These images are grouped into z-stacks, which are in turn used to create a number of projection images. These projections are then searched for pollen and annotated. For the annotation process, several classical processing methods are explored. Annotated images are split into tiles, keeping only the tiles with pollen. Once pre-processed, these images will be formatted for each model and split into 5 folds, such that each has a different train and validation split. A test split is similarly created using different tree scans (to mimic real world use).

3.1 Data collection

The data we used for this project has been provided under supervision of Nemi Dorst from Naturalis, an experienced bioinformaticist working on a bigger project around pollen. The single species samples were collected from trees in Leiden, the Netherlands, by collecting catkins during their flowering season of 2024. Flower parts of these catkins were opened using sterilized tweezers, and pollen was sprinkled onto glass microscope slides. As opposed to pollensniffer scans, pollen in a catkin scan are all from a single tree and contain little debris of any kind, making them ideal as training data.
When preparing and scanning the slides of each pollen, it is important that the settings of the image and staining are done the same. If there are differences in the final pollen slides aside from the pollen themselves, the training model may use these factors to differentiate the species, introducing a problematic feature bias. Although this problem could be avoided by placing multiple pollen species on the same slide, this would mean they would have to be annotated and classified by hand, which was outside the scope of this project. As such, effort must be taken to ensure the method by which samples are obtained is stable and precise. Each sample in our study was stained with safranine, which binds to the glycoproteins on the outer layer(exine) of pollen to better distinguish them from other debris like fungal spores and particulate matter. These samples were examined using a Zeiss Axioscan 7, a brightfield microscopy scanner. The scanner takes a prescan picture (x10, 0.95 Numerical Aperture), which it uses to autofocus the samples and finds the 20 best z-stacks to get good visuals, meaning little work was necessary by hand. Each stack is then captured at x40 magnification with 8 bit format and 3 colour channels.

3.2 Model Preprocessing

The scans (in file type .czi) are converted into slides of size 2056 by 2464 pixels (in file type .tiff) for processing. For each location in the lateral plane, images from twenty different depths are created to make sure all pollen are in focus. Each z-stack of twenty slides is then projected onto a single image, both to save space and since this has been shown to work better than training a model on all images [Konijn et al., 2025]. This is most likely

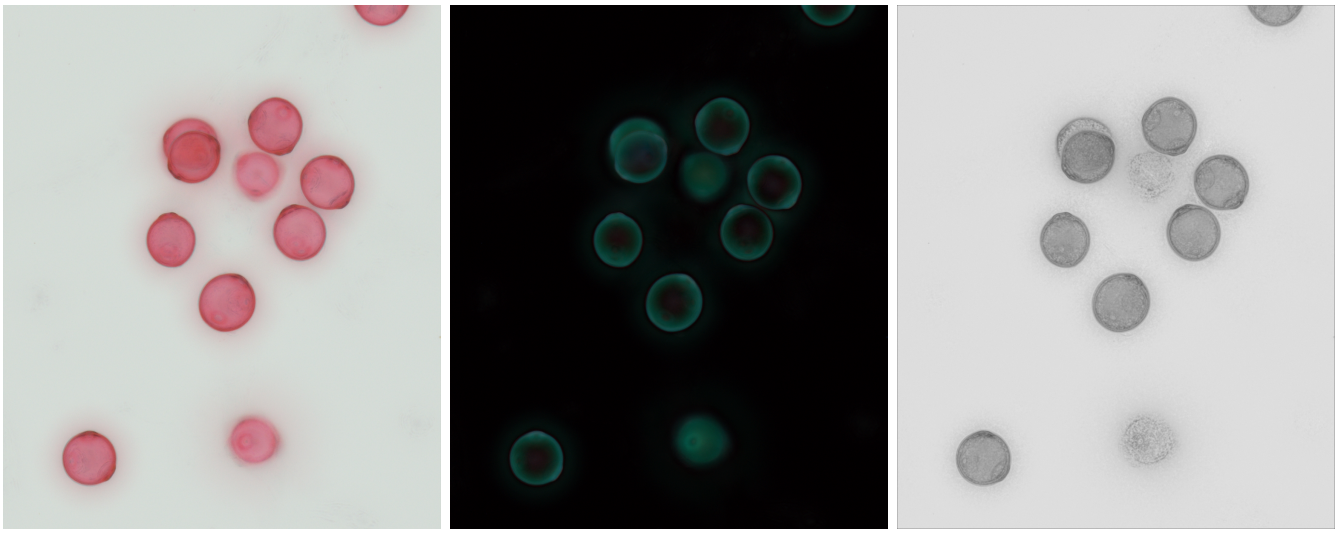


Figure 2: Comparison of the same Betula pendula pollen image stack using different projection methods. From left to right: minimum projection, standard deviation, extended focus projection.

because most stacks will not have the pollen in focus, and those that do will not contain more information than the projection image. For this project, three projections are made using the Standard Deviation (std), Minimum (min), and Extended Focus (ext) methods. Standard deviation examines the differences between pixel intensities throughout the stack, increasing in brightness where more change is found. Minimum intensity takes the lowest pixel value for each pixel in the stack and uses that for the projection (since the background is brighter than the objects, this will usually be the sharpest slide of each object). Lastly, the Extended Focus projection detects edges using Sobel filters on each layer, then uses the most in-focus layer for each pixel via maximum edge strength. It then blends these selected regions using a blurred altitude map to produce a single image where all parts appear in focus. Each of these projection methods have been chosen since previous research has shown their success when used in single pollen classification [Polling et al., 2020, Li et al., 2023]. Before loading these images into the model, we implement a tiling algorithm to cut the images to smaller sub-images of 1024 by 1024. The image size is lowered to lower computational costs. A drawback of tiling is that it increases the number of images and the chance that we lost pollen by cutting them between multiple images. We attempted to offset this by adding a conservative 50 pixels of overlap to the tiles; adding more would introduce double counting.

3.3 Bounding Box annotation

Having suitable images for our training, labels are still required before models can be trained for the detection task. Labelling for detection models requires two aspects for each object instance; the class of the object (usually denoted by an integer) and the coordinates of the box which denotes the object location on the image. Since each catkin slide scan contains one species (and it is known which one), this works well as our classification for the labelled data. This leaves the challenge of creating the bounding boxes around each pollen grain. Since the number of instances is very high, labelling each pollen by hand seems very time consuming. Moreover, this process needs to be repeatable for future inclusions of new species or additional samples. Exploration into classical, rule-based, image analysis methods for pollen detection could provide a successful alternative pipeline for creating ground-truth data. Catkin scans work better than pollensniffer samples, as they are relatively clean and contain only a single species. If classical filtering can generate sufficiently accurate annotations, these could then be used to train deep learning models that generalize to more complex cases, such as pollensniffer samples. More importantly, inspecting these methods is of vital importance into the broader search for the most accurate pollen detection; a perfect classical method could relinquish the need for an algorithm-based detection functionality entirely. However, achieving such performance on pollensniffer samples is implausible, since these are substantially noisier and may include abnormal particles, similar non-target pollen types, and other unknown sources of interference.
To create classical pollen detection method for the catkin scans, multiple filtering methods have been attempted to obtain single pollen bounding boxes. In this segment we will analyse and compare multiple methods and

choose the one most successful. Before each method, steps are taken to maximise their performance and lower computational cost. To start off, we threshold and segment each connected component to get our starting objects. Doing so we save processing time on background or debris. However, the slide objects can still encompass multiple particle types, from intact pollen grains, broken pollen fragments, large debris, clusters, and air bubbles. The methods we thought of to ascertain the single pollen from these slides, including a feature support vector machine (SVM), watershedding, Hough-circle filters and template matching. Each method was implemented, as can be seen in Figure 3. For usable methods, performance was measured against a dataset with 50 images per species.

- **Feature SVM + Watershed:** Using a feature SVM, objects can be separated into usable and unusable categories. This is done by inspecting the masks of the objects, extracting shape and size features, and filtering them. This first requires training, which is done with around 300 samples by hand. This method excelled at finding pollen from non pollen. However, this method had difficulty with clusters and pollen on the sides of images. Clusters were common in our training data, and so if this method was taken, a separate method was necessary to divide the pollen in clusters. For this, watershedding seemed the most obvious choice. Watershedding could be used to divide the circular objects by taking the skeleton by zones of influence(SKiZ) and splitting at the min distance points. Unfortunately, this would not work with heavily overlapping pollen. Moreover, although the SVM was very successful at finding individual pollen, it had trouble differentiating clusters from debris. Therefore, this method was abandoned.
- **Template Matching:** Template matching would be good at finding objects that differ little depending on orientation with a not too variable size. For our template, a separate template was attempted for each species. Each template was a combination of around ten to fifteen pollen images scaled to their mean size, flipped in all directions, and overlapped to create pixel averages. If objects differ depending on direction or size in the image, multiple templates are needed for each pixel, to make sure the matching format is found. Fortunately for pollen, only varying size is a concern, so four representative sizes of the template were used. During template matching, each pixel in the detected object area is scanned using a normed correlation coefficient, so that slight differences in pollen stain strength(changing intensity and contrast) do not affect the filter too much.
- **Hough Circle Filtering:** Another option is Hough filters; these excel at finding round contours. These could search for circular objects in further processed min projections, which is helpful since the darker edges of each pollen are clear even when overlapping on the bright field images. We can classify the filters radius size and confidence strength to fit the pollen size and intensity. To make sure we only accept clear pollen for classification, we additionally run a Sobel filter to highlight the edges and overlap this with the inverse of the min projection (so the grains are high intensity). This will filter out most small and non circular objects, as well as split clusters. Finally, to remove objects like thick air bubble edges, we multiply object RGB values with themselves and filter out all very dark objects, since pollen are usually only red (like Betula pendula) or purple (like Callitropsis nootkatensis).

Pollen	**Metric**	**Hough Circle**	**Template Match**
Betula pendula	Precision	0.9636	0.8984
	F1-score	0.9464	0.9385
Callitropsis nootkatensis	Precision	0.8966	0.0321
	F1-score	0.8525	0.0388
Chamaecyparis lawsoniana	Precision	0.8448	0.4925
	F1-score	0.7656	0.4818
Total	Precision	0.9017	0.4743
	F1-score	0.8548	0.4864

Table 1: Comparison of precision and F1-score between Hough circle filter and template matching methods for each pollen types. As you can see the Hough circle filter outperforms template matching.

Out of these, the hough circle filtering proved most successful. Looking at the metrics from Table 1, we can see the results of the Hough and template methods on around 50 images per species. These images were hand-picked to be difficult with debris and clusters of pollen. We found that the Hough method had a precision of

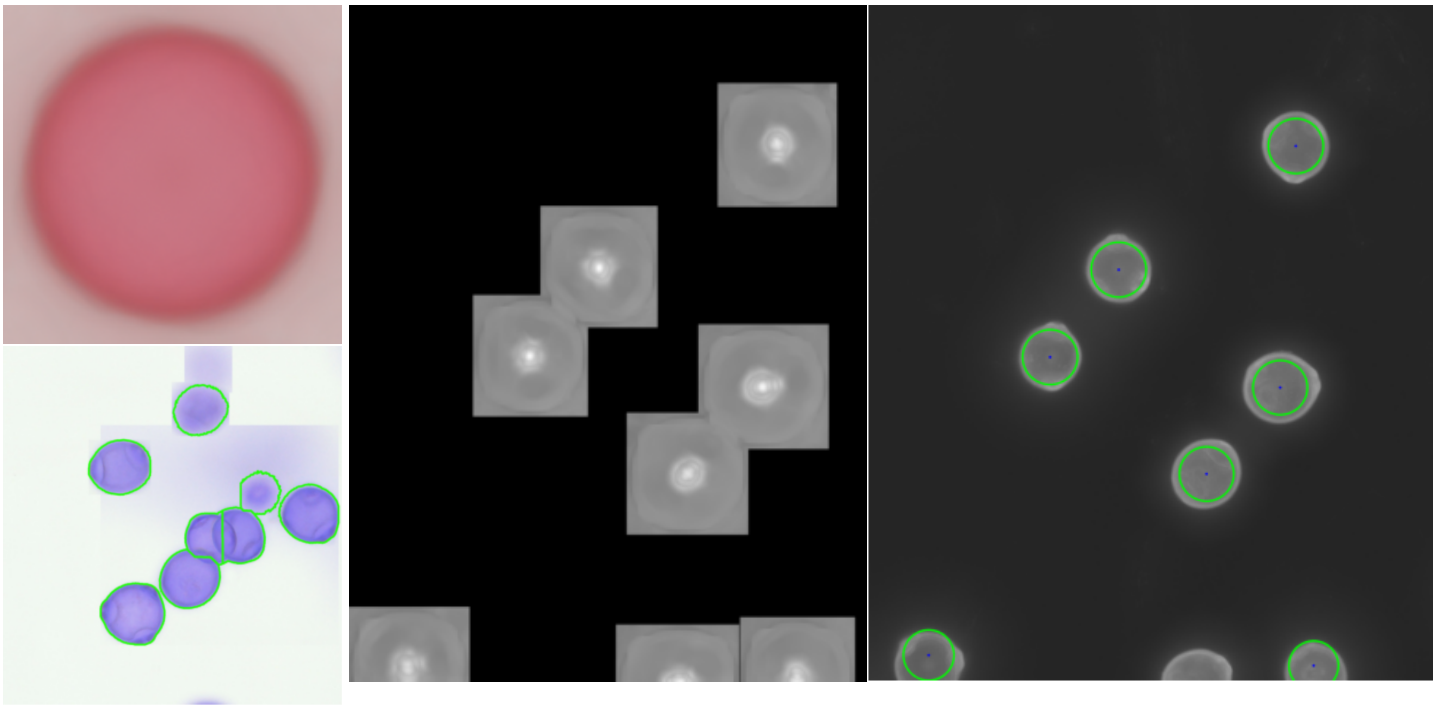


Figure 3: Visualisations of image processing methods to create pollen bounding boxes. Top Left: the Betula pendula template used for template matching. Bottom Left: watershed segmentation. Middle: template matched heatmap, using the maximum scoring template per pixel. Right: Hough-based circle detection, showing the best radii and center-points.

90 percent and performs even better when we used easier samples. This was found usable for network model ground truth data, but not good enough to forgo the need for a detection network entirely. On the contrary, it provides a compelling argument for network model-based detection; that classical filtering cannot achieve the desired annotation accuracy on catkin scans, let alone pollensniffer scans.

3.4 Pollen Counting Networks and Optimisation

To analyse one-stage object detection performance on pollen grains, three benchmark models were selected: YOLOv8, DINO (Swin-DETR), and RetinaNet(with Resnet-50 backbone). All three were implemented using their official repositories and pretrained on the COCO dataset. These were then trained using the pre-processed, labelled data described in the preceding sections. All experiments were performed on a Linux Ubuntu system with two GeForce RTX 2080s, each having 8GB of memory. All experiments were run with python 3.10 using Cuda 11.6 compatible torch. Automatic mixed precision was enabled throughout the training of each model.

3.4.1 Yolov8

YOLOv8 is the first one-stage detector for the pollen detection task comparison. It is released by ultralytics as a follow-up of previous YOLO versions and outperformed them on mAP scoring of the COCO Dataset while remaining lightweight [Jocher et al., 2023]. The model we are using is the pretrained YOLOv8s architecture from the ultralytics library. The 's' in Yolov8s indicates small, having around 9.4 million parameters. For model parameters, most of the model default parameters were kept for the preliminary test. Standard parameters include a stochastic gradient descent optimiser, a momentum of 0.937 (meaning past gradients slightly impact the current update), a weight decay of $5 \cdot 10^{-4}$(to penalise very large weights), and 3 warm-up epochs with added momentum of 0.9 (so that early epochs do not have too much effect). YOLO uses a combination of three loss values for its optimiser, the objectness loss, bbox regression loss and class prediction loss. Combined, these serve well to estimate how well the model is doing. Since it is usually used for training with less instances, the models initial learning rate was lowered to a more stable 0.005 from its standard 0.01 and cos_lr, a scheduler which adjusts the learning rate to follow a cosine curve over epochs, was enabled. The model was pretrained on the COCO Dataset, except for the model-head, which were altered to have only three classes. Furthermore,

due to a combination of having large images(1024, 1024) and our computational limits, the batch size was set at 4. Lastly, each fold was trained for 100 epochs, with an early stoppage patience of 20 epochs based on the mAP50-95 improvement. This happened relatively frequently, stopping usually around epoch 80. Model training of all five folds on took around 8 hours.

3.4.2 DINO

The DINO (DETR-Swin) architecture was used to evaluate the performance of a transformer-based approach for pollen detection and classification. The model was implemented with a Swin-T-224-1k backbone and trained for 100 epochs using the AdamW optimizer. This backbone was pretrained on the COCO dataset. An initial learning rate of $5 \cdot 10^{-5}$ was used for all parts of the normal, lowered from its default $5 \cdot 10^{-4}$ but keeping the backbone the same. The backbone learning rate is usually kept lower since this part of the model is usually pretrained while the head is often replaced. However, since our dataset differs from the standard COCO dataset, having only objects of a single limited size, we altered both to have similar learning rates. Furthermore, a warm-up period of 5 epochs was used, as well as a learning rate decay after 50 epochs. Training was conducted with a batch size of one image per GPU, using two devices for parallel training, yielding an effective batch size of two. Model training of all five folds on took around 2 days.

3.4.3 RetinaNet

RetinaNet was implemented as the second one-stage object detection model in this study. The network was instantiated using the `retinanet_resnet50_fpn` architecture from the PyTorch `torchvision` library, which couples a ResNet-50 backbone with a Feature Pyramid Network (FPN) (Lin et al., 2017). In this work, the weights for the backbone were pretrained on the ImageNet dataset, while the detection head was configured to match the number of pollen classes (plus one background class), yielding an output layer with four units. The model was trained for 100 epochs with a batch size of two images per iteration, using stochastic gradient descent (SGD) with a learning rate of 0.001, a momentum of 0.9, and a weight decay of $5 \cdot 10^{-4}$. A StepLR scheduler was used, reducing the learning rate by a factor of ten every 30 epochs. The loss comprised the built-in Focal Loss for classification and Smooth L1 loss for box regression as implemented in the torchvision RetinaNet model, making it robust against imbalanced class distributions and smaller visual differences between pollen species. By training folds separately on each GPU, model training of all five folds took around a day and a half.

3.4.4 Optuna

Aside from our models, a parameter-optimisation method was needed for experiment 3: model Optimisation. To tune parameters for the chosen model of the final experiment, multiple strategies are possible. Formerly, grid search and random search are easy to implement, effective methods. However, since these do not alter their search strategy based on earlier trial results, they can waste time looking in search areas of the search space where results are lower. Instead, this experiment opted to tune the model using the Optuna framework. Within a search space of chosen parameters, Optuna is requested to make a selection for each trial with a set number of trials. Trials results are recorded, and successive trials are chosen with parameters that seek gradual improvement while exploring the search space. Furthermore, it allows us to prune results that perform poorly after the first folds, saving time. This pruning ability is particularly valuable in our context, since individual training runs are computationally expensive. Each trial, it samples the parameters in the location it estimates the best performance, using a chosen estimator like the Tree-structured Parzen Estimator(TPE) [Akiba et al., 2019]. It works by splitting all previous trial results into a list of good and bad trials and tries to create empirical condition densities for each group. It then samples a number of points from the good group distribution, from which it picks the one which takes the trial with the highest ratio of the good group distribution divided by bad group distribution using an acquisition function. Using this method, we explore the search space in a more directed manner than a method like grid search or random search, as each trial should perform better than the last.

3.5 Model Evaluation

Aside from training, the ground truth data can be used to evaluate how well the model performs. Since the model must predict both the correct class and accurate bounding boxes, multiple evaluation metrics are

required.
Firstly, the Cross-Entropy (CE) loss assesses the model's ability to predict the correct class when an object is detected. Meanwhile, bounding box regression (BBox) and the Generalized Intersection over Union (IoU) evaluate the spatial accuracy of the predicted bounding boxes. BBox does this by computing a smoothed distance between the predicted and ground truth coordinates, while IoU measures the overlap between predicted and actual boxes. IoU is calculated as the ratio of the area of intersection to the area of union and introduces an additional distance penalty when the boxes do not overlap. These metrics collectively provide a comprehensive view of the model's learning progress and performance. During training, they are typically used to monitor convergence per epoch and to identify if the model is trapped in a local optimum.
For final evaluation, a combination of precision (P), recall (R), and Average Precision (AP) is used, particularly the metrics mAP50 and mAP50-95 [LeCun et al., 2015]. Precision measures the accuracy of positive predictions, while recall measures the model's ability to detect all relevant objects. These are computed as follows:

$$P = \frac{T_p}{T_p+F_p}, Re = \frac{T_p}{T_p+F_n}$$

where T_p is the number of true positives, F_p the false positives, and F_n the false negatives. To balance precision and recall, the F1-score is used, defined by the harmonic mean of the two:

$$F1score = \frac{2*P*Re}{P+Re}.$$

Average Precision (AP) is obtained by computing the area under the precision-recall curve (AUC-PRe). Since both precision and recall require a threshold to determine whether a prediction is correct, a constant IoU threshold is applied (commonly 0.5 for lenient evaluation or 0.95 for strict scoring). The mean Average Precision (mAP) is then calculated by averaging the AP scores across multiple confidence intervals:

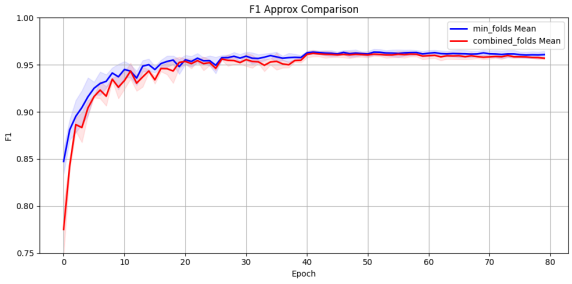
$$mAP = \frac{1}{n}\Sigma_{i=1}^{n} AP_i.$$

For mAP50–95, AP is averaged over 10 IoU thresholds from 0.50 to 0.95 with 0.05 increments. Although IoU is not directly related to the task of counting objects per image, it remains a useful indicator of how accurately the model localizes the pollen grains.
Data for the validation set must be kept separate from the training set, so that the validation set can check for bias from training that is not actually useful when trying to create predictions for the real problem. Likewise, when a model is hyper-optimised, the act of tuning the model on the validation set can introduce bias for this subset. For this reason, an independent test set is likewise reserved from both the train and validation set for unbiased performance assessment. Lastly, to increase the robustness of the evaluation, 5-fold cross-validation is employed. This involves dividing the dataset into k subsets. The model is trained k times, each time using k-1 subset for training a different remaining subset for validation. It improves validation generalization by increasing the amount of data the model is evaluated on, whilst also testing how robust the model is to different training data. When building a final model, the best performing fold may be selected. However, for evaluation purposes, the results of all folds are averaged to estimate overall model performance. Using 5 or 10 folds is a standard practice in AI and machine learning, offering a balance between reliable estimation and computational efficiency [Berrar et al., 2019].

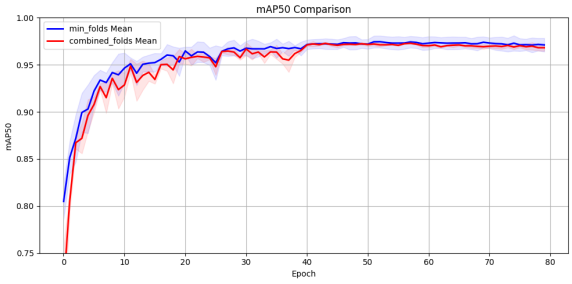
4 Results

4.1 Experiment 1: Projection Input Comparison

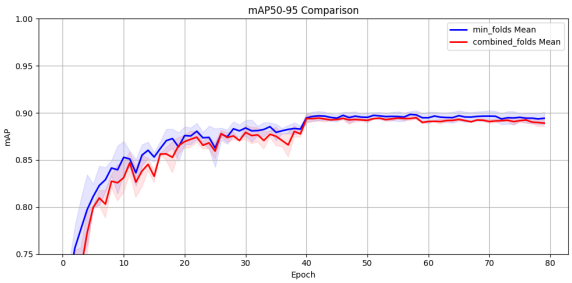
This experiment investigates the impact of different preprocessing methods on model detection performance. Since most models used on 3 channel input images, our preprocessing was matched to this so the models could be pretrained. We compared an approach using the coloured minimum projection(min) against the approach using the greyscales of the minimum, standard deviation and extend focus projections(combined). These two methods were chosen since they appear most viable, the min projection gives the best impression in colour differences to our visuals, while the greyscale channels combined have been chosen in earlier papers to be reliable input formats [Li et al., 2023]. Combined enhances information gained through the depth of the z stacks while the min projection increases information of the objects like how well they stain. The goal is to assess which preprocessing technique best improves detection precision, recall and map scores, and highlights pollen grains correctly. Model choice is presumed arbitrary for this experiment, and DINO was chosen as the originally expected best performing model.



(a) F1-score comparison



(b) mAP50 comparison



(c) mAP50-95 comparison

Figure 4: Comparison of averaged DINO performance using either the coloured minimum projection (blue) or the combined greyscale projections (min, std, and ext) (red). The results are very similar for both the F1-score and the mAP50 and mAP50-95, but the minimum projection performance converges faster and slightly outperforms the combined projection.

Looking at the figures of Figure 4, it would seem min projection outperforms the combined projections with marginal accuracy improvement. Although the results between the two input types are minimal, the fold-averaged best mAP for min is around 0.727% better than combined. It is unusual the two are so similar, given that the difference between input types seem both of notable importance and to be very different information. Different since the colour in the min input clarifies staining strength of the sharpest layer, while the combined projections input gives more depth information.

4.2 Experiment 2: Model Comparison

In this experiment, we compare the three benchmark models against one another in their ability to accurately detect and classify pollen grains. Each model is trained and evaluated using the coloured minimum projection since it was the better performing pre-processing method from the first experiment. When comparing results, the metrics of interest are the detection accuracy (mAP), precision, and recall. To compare not just the final score, but also training behaviour, we will examine these scores after each training epoch. For this evaluation,

we train each model on five folds, which are averaged before looking at the final comparison. Lastly, since DINO did not seem to improve much after epoch 45, and is computationally slow to train, it was opted to cap its training at epoch 80 instead the 100 epochs originally planned.

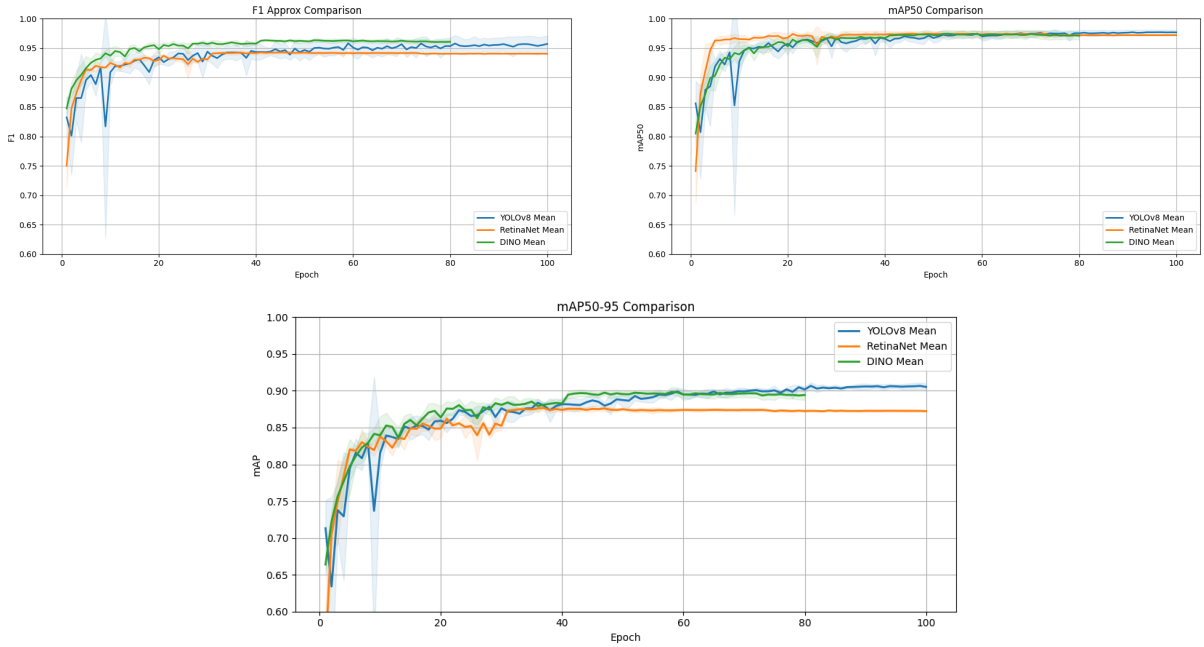


Figure 5: Visualisations of model performances averaged over all 5 folds. From left to right: the f1-Score, the mAP50 the mAP50-95 and the loss. You can see YOLO outperforms the other models at the end of training for all non loss metrics. Looking at mAP50, you can see Retina is fastest to converge.

Looking at the results of the models in figure 5, we can see RetinaNet has by far the fastest convergence, however, seems to flatten out relatively quickly. DINO improves a lot the epochs after its learning rate drops (at epoch 40) but seems to get stuck in a local optimum and not improve much once settled. YOLO performs slowly but relatively well, slowly converging even at epoch 100. This may have to do with the built-in cosine learning rate drop feature. It however does have higher variance between folds, and a propensity to spike at lower epochs. Yet at the end the average score for both mAP50 and mAP50 to 95 is higher with more than a percent compared to the other two models. For this reason, it was used in the last experiment.

4.3 Experiment 3: Model Optimisation

The final experiment applies parameter optimization to the best-performing model and preprocessing technique in order to further refine performance. In this experiment, the strategy chosen for this task is tuning using the Optuna framework, with 20 trials of training. The objective we used for parameter tuning in this experiment was mAP50, since the exact overlap of the boxes was not as much of a concern. The parameters chosen to tune in this optimization problem were the following:

- the initial learning rate (between 1e-5 and 1e-2)
- the weight decay (between 1e-6 and 1e-2)
- the optimizer (stochastic gradient descent or adamW)
- and the model size (either YOLOv8s or YOLOv8m)

Within this search space, Optuna is requested to make a selection for each trial. When this trial is either finished through pruning or after training on all 5 folds, the mAP50 is marked within the space so that a better option can be chosen for the next trial. This is repeated for 20 trials. Once done with testing, the best performing model is chosen and assessed on the test set, this is the final performance metric for real world use of our optimal single-use model.

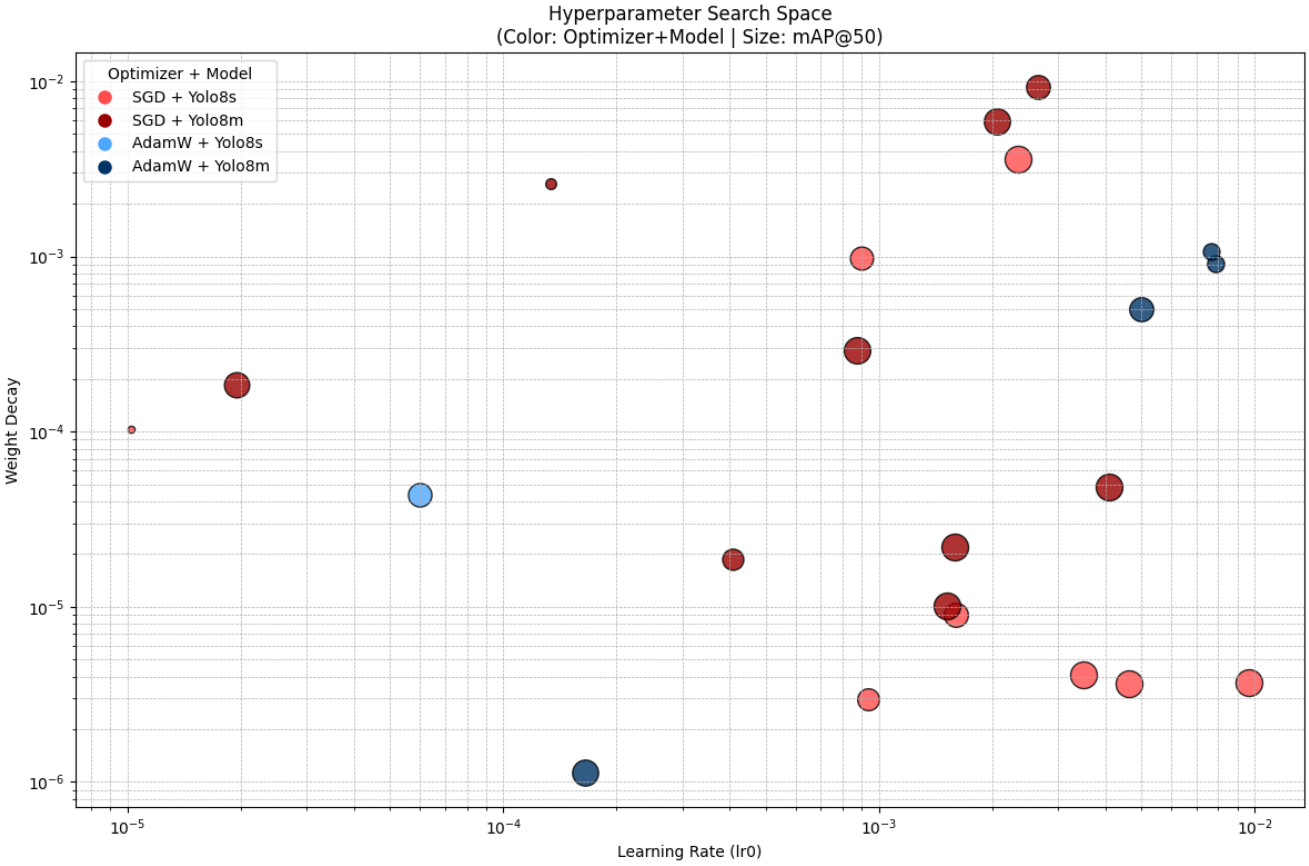


Figure 6: Visualisation of experiment 3 search space exploration. Each dot represents a trial. Their location and colour and shade denote parameters, while dot size denotes performance.

We can see Optuna exploring the search space in Figure 6. Looking at the performance, we can see that the medium model suffers at a high weight decay with low learning rates. This makes sense, since larger models have a lot of weights to train meaning that a low learning rate and weight decay slowing training the model will have trouble converging to a good optimum. Furthermore, we can see the results are relatively similar in the search space, and as such Optuna explored a very wide area. This tells us that the chosen model, YOLOv8, is much more robust to parameter changes than for instance a model like DINO. Regardless of the low performance differences, the best model was identified by examining their average mAP50 scores. The results of our best optimised model on the validation set are shown in Figure 7. The best combination of parameters for our validation set are the following:

- initial learning rate: $2,35 \cdot 10^{-3}$,
- weight decay: $3,58 \cdot 10^{-3}$,
- optimizer: *stochastic gradient descent*,
- model: YOLOv8s

Using these parameters, the model reached an average mAP50 of 0.965 (with a standard deviation $5,42 \cdot 10^{-3}$) of and an average mAP50-95 of 0.846 (with a standard deviation $2,46 \cdot 10^{-2}$). These are very good averages, with the low standard deviations showing the results are reproducible and reliable. Evaluating this model on our test set we get Table 2. If we take the best fold, this gives us a final F1 score of 83,2%, with a mAP50 of 85,6%. Although reasonable, this is notably worse than the validation set scores.

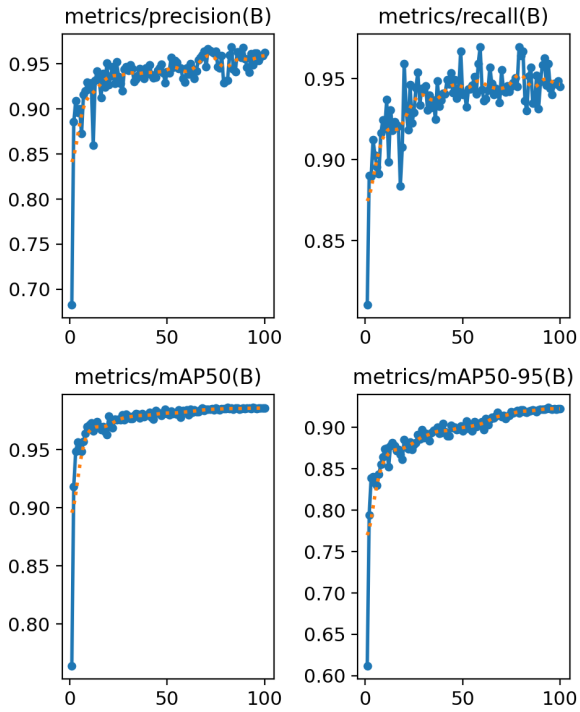


Figure 7: Model performance on validation set during training using optimal parameters. All graph x-axis show number of epochs trained. Top-left depicts Precision, top-right depicts Recall, bottom-left depicts mAP50 score, bottom-right depicts mAP:50-95 score.

Fold	**Precision (P)**	**Recall (R)**	**F1-score**	**mAP@50**
0	0.816	0.603	0.6935	0.670
1	0.864	0.803	0.8324	0.856
2	0.709	0.786	0.7455	0.837
3	0.732	0.789	0.7594	0.825
4	0.617	0.752	0.6778	0.775
Mean	0.7476	0.7466	0.7471	0.7926
Max	0.864	0.803	0.8324	0.856

Table 2: Test set results after parameter tuning across 5 folds.

5 Discussion

Looking at our results, we can make the following observations. In the first experiment, it is notable that the two input forms both seem to perform similarly. Looking at the different genera of pollen, we observe that their colours differ slightly under staining, which led us to expect that the minimum colour projection would outperform the combined greyscale projections (minimum, standard deviation, and extended focus). Yet, the difference in performance was almost negligible. This suggests that the combination of projections provides an advantage roughly equal in value to the benefits of colour information alone. One potential explanation is that the greyscale projections enable better texture recognition, offering additional surface information of the grains that aids classification. Another possibility is that we underestimated the utility of the other projection types, standard deviation, and extended focus, which may capture complementary information not present in the minimum projection alone. However, further experimentation would be required to assess the value of each individual projection type. Another avenue worth exploring is the combination of all three projection types with colour, resulting in nine input channels. This could potentially enhance performance by offering a more comprehensive input representation. However, such an approach would require either adding an extra layer before the model's existing architecture, which is computationally expensive, as the first layer has the most weights, or modifying the first layer itself, which would make pretraining infeasible for this layer.

Before we continue to the second experiment, a note on our ground training is in order. Although our development of automated annotation using the Hough transform method was effective, its use for catkin scan ground truths slightly damages the experiments accuracy. Not only by altering the model's accuracy, but the difference between hand annotation and the filter has an effect on our test results, and as such can be seen as a critique on the results validity. Manually annotated pollen would result in higher-quality ground truth data and, consequently, improved model performance and more accurate test results. Alternatively, further optimisation of the Hough method could eventually produce a viable detection approach, in combination with other techniques. Looking at the Hough detection rate in the methods section we can see it suffers more for certain pollen types than others. If steps are taken to improve the Hough circle filter method, per species refinement would be advised.

In the second experiment, we compared three model architectures to determine whether certain structures might outperform others in this task and whether any general conclusions could be drawn about architecture performance. Each model was trained in a 'standard' setting, as this was assumed to provide optimal general performance. Interestingly, all three models achieved similar accuracy levels, with no model outperforming the others by more than 1.5 percentage points. This suggests that within the space of deep CNN one-stage models, model choice is not of critical importance so long as the model has sufficient capacity to adapt to the task-specific features.

From the training graphs of this experiment, we can draw additional insights. First off, RetinaNet shows rapid convergence. While the focal loss function is an expected cause, it is also possible that the difference stems from the dataset used for pretraining, ImageNet rather than COCO, unlike the others. However, the most plausible explanation could be that RetinaNet had the highest initial learning rate since it did not use warm-up epochs, meaning training could start faster.

DINO, the only transformer-based model, demonstrated notable stability. Aside from the initial warm-up epochs, which were necessary to prevent exploding weights, it showed the lowest standard deviation and the most consistent performance. Nonetheless, the model was run with a small batch size due to computational constraints, and the learning rate decay after 40 epochs may have been too steep. We speculate that with a larger batch size, more gradual learning rate decay, and similar tuning to what was done for YOLO in the third experiment, DINO could potentially outperform the other models.

This brings us to YOLO. YOLO achieved the best performance overall, converging to the highest accuracy. An explanation for this could be that the task does not require complex feature extraction such as contextual awareness or orientation detection. In such a case, a lighter model with fewer weights may have an advantage. Interestingly, despite outperforming on all performance metrics, YOLO's test loss scores were worse compared to the other models. This may suggest the other models are overfitting on the loss, explaining why YOLO could have better behaviour.

In the third experiment, we focused on optimising the YOLOv8 model to improve its accuracy. For this, we implemented the Optuna optimiser. The optimal parameter settings were found to be a medium learning rate and weight decay (between $1 \cdot 10^{-2}$ and $1 \cdot 10^{-3}$) , stochastic gradient descent as its optimiser and a small size instead of medium. The model size is in line with our results from experiment 2. The metrics from running the model with these parameters result in a validation precision of 96,3% and an mAP50 score of 98,6%. This level of accuracy seems acceptable for real-world application. However, when evaluating it on the test

set, a F1-score of 83,2%, with a mAP50 of 85,6% is significantly lower. We theorise this is not because of overfitting, as even without parameter tuning, which examine the validation set, the results were not this low. A likely alternative explanation is that the test set uses data from different scans. These scans have pollen from different trees and slight changes in staining.
Lastly, when comparing our results to those achieved by two-stage classification algorithms, of which the best had a classification accuracy of 99.4% ($\pm$0.002) on other pollen, it is important to note that our model also includes object detection. This means that even with substantially lower performance, one-stage could outperform two-stage models overall, since those still require an additional model which adds separate detection errors.

6 Future Work

Evaluating this paper's methodology and experiments, recommendations can be made to guide future studies: First, cropping seems of considerable importance when determining the accuracy of not just the models, but also the classically automated annotation. A relatively large image size was used to attempt to offset this, but pollen that are cut out of the picture make it hard to make good estimations as to whether any algorithm is making error in its recognition, or simply setting a different threshold as to how much of the pollen grain must be in the image. When sampling the Hough filter for errors, none were found, yet the results were different to hand annotation due to a considerable number of pollen being on the edges. Aside from using larger tiles(sub-images), a post processing method was thought up to solve this by using a combination of higher overlap and noting the total scan locations of all pollen predictions using tile metadata. Then, pollen with too similar locations can be filtered out so that no pollen is counted double, and edge pollen can be disregarded annotation. However, this was outside the scope of this study as the data used did not have the necessary metadata.
Second, our models were trained and validated using samples from a maximum of two trees per species. Initially, the plan was to include three trees per species, but the time-consuming acquisition and scanning process limited this. As a result, the model may underperform when exposed to samples from different trees due to unaddressed intra-species variance. Future work should prioritise hand-annotating pollen from more trees to create a more robust and representative training set. Lastly, we were unable to train models using actual air-sniffer samples. These samples often contain multiple pollen types, making it infeasible to generate clean, classified bounding boxes using our current methods. Furthermore, environmental factors such as petroleum jelly, covering tape, or degradation in the sample may alter the visual appearance of pollen. Although methods of preprocessing like deconvolution or model data augmentation like blurring could be explored, this will still impact model performance. This poses a potential challenge to generalisation and should be studied further.

7 Conclusion

Airborne pollen monitoring can currently be conducted in three steps: pollen sample acquisition, pollen sample preparation and staining, and manual microscope counting. This final step is often slow, extremely labour-intensive and, for certain species such as those in the *Cupressus* genus, impossible due to morphological similarities. These challenges restrict the method's scalability and restrict if from certain applications, like real-time forecasting. In this paper, we introduce an alternative method using AI, replacing the last step with a quicker option. This option requires the samples to be scanned and imaged, so that these images can be processed and fed into a model. This saves researchers time and may improve the pollen detection, identification, and quantification process. Specifically, in this research we look and compare the possibility of using one-stage deep learning models to both recognize and classify distinct species of tree pollen.

Before exploring the model space, a comparison was done of preprocessing methods and recognition methods for generating ground data. Earlier works showed that taking projections of depth stacks of scans improved model performance and saved computational resources. In this paper, two options were explored, both with 3 channel image inputs: a combination of minimum, standard deviation, and extend focus greyscale projections and a RGB coloured minimum projection. Models gain an advantage when given coloured projections, however not by a major amount, and it might be worth pursuing combinations of multiple coloured projections, since the standard deviation and extend focus projections seemed to contain useful information for the models as well.

Furthermore, to gain a basis for recognition comparison and ground data filtering method, we explored multiple methods for single pollen segmentation. Objects were filtered with multiple forms of segmentation, feature support vector machines, watershedding, Hough circle filter, and combinations of these methods. A segmentation followed by a Hough circle transform to find pollen centre points was proven most successful, having an accuracy of 90% on low debris catkin scans. Although useful for generating the bounding boxes for our one-stage models, these detection methods would not work on images with high counts of other debris and round pollen.

To get a good representation of the one-stage model space, multiple different models are compared, each with different advantages: YOLOv8s, RetinaNet with a Resnet-50 backbone and DINO, with a Swin backbone. YOLO, a popular and fast model without anchor boxes used for real-time processing. RetinaNet, a feature pyramid network with focal loss to address class imbalance, and DINO, a visual transformer that scores high on the COCO dataset and does not rely on non-maximum suppression. These take longer to train but are noted to be very reliable once optimized. Of these, our results showed that YOLOv8 performed best, however it can be stated that the difference in performance was minor. This means that (given the architecture can fit to the problem) no clear model architecture would outperform the others, and model choice was of lower importance than for example parameter optimisation. For YOLOv8, the best parameters were found within search space using Optuna hyper-optimisation, with a mAP50 of 98,6%. However, is performed notably worse on a new scan from different trees of the same species, getting a F1-score of 83,2%, with a mAP50 of 85,6%. This suggests that a broader, more diverse training dataset could very much improve the final results. Although improvement is still advised, we are confident this model could be implemented for pollen counting for most tree pollen in real world use.

References

[Akiba et al., 2019] Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631.

[Berrar et al., 2019] Berrar, D. et al. (2019). Cross-validation.

[Buters et al., 2024] Buters, J., Clot, B., Galán, C., Gehrig, R., Gilge, S., Hentges, F., O'Connor, D., Sikoparija, B., Skjoth, C., Tummon, F., et al. (2024). Automatic detection of airborne pollen: an overview. *Aerobiologia*, 40(1):13–37.

[Carion et al., 2020] Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. (2020). End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer.

[de Weger et al., 2020] de Weger, L. A., Molster, F., de Raat, K., den Haan, J., Romein, J., van Leeuwen, W., de Groot, H., Mostert, M., and Hiemstra, P. S. (2020). A new portable sampler to monitor pollen at street level in the environment of patients. *Science of the total environment*, 741:140404.

[del Pozo-Banos et al., 2015] del Pozo-Banos, M., Ticay-Rivas, J. R., Alonso, J. B., and Travieso, C. M. (2015). Features extraction techniques for pollen grain classification. *Neurocomputing*, 150:377–391.

[D'Amato et al., 2017] D'Amato, G., Vitale, C., Sanduzzi, A., Molino, A., Vatrella, A., and D'Amato, M. (2017). Allergenic pollen and pollen allergy in europe. *Allergy and allergen immunotherapy*, pages 287–306.

[Endo et al., 2024] Endo, K., Hiraguri, T., Kimura, T., Shimizu, H., Shimada, T., Shibasaki, A., Suzuki, C., Fujinuma, R., and Takemura, Y. (2024). Estimation of the amount of pear pollen based on flowering stage detection using deep learning. *Scientific Reports*, 14(1):13163.

[Gisler, 2021] Gisler, A. (2021). Allergies in urban areas on the rise: the combined effect of air pollution and pollen. *International Journal of Public Health*, 66:1604022.

[Jocher et al., 2023] Jocher, G., Chaurasia, A., and Qiu, J. (2023). Ultralytics yolov8. 2023.

[Khan et al., 2020] Khan, A., Sohail, A., Zahoora, U., and Qureshi, A. S. (2020). A survey of the recent architectures of deep convolutional neural networks. *Artificial intelligence review*, 53:5455–5516.

[Konijn et al., 2025] Konijn, T., Bijl, I., Cao, L., and Verbeek, F. (2025). Analysis of 3d urticaceae pollen classification using deep learning models. *arXiv preprint arXiv:2503.07419*.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

[Kumar et al., 2023] Kumar, S., Arif, T., Ahamad, G., Chaudhary, A. A., Khan, S., and Ali, M. A. (2023). An efficient and effective framework for intestinal parasite egg detection using yolov5. *Diagnostics*, 13(18):2978.

[LeCun et al., 2015] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

[Li et al., 2023] Li, C., Polling, M., Cao, L., Gravendeel, B., and Verbeek, F. J. (2023). Analysis of automatic image classification methods for urticaceae pollen classification. *Neurocomputing*, 522:181–193.

[Liao et al., 2024] Liao, W., Luo, L., Wang, C., and Zhang, C. (2024). Cell dino: End-to-end cell segmentation and tracking with transformer. In *2024 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 3491–3494. IEEE.

[Lin et al., 2017] Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2017). Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988.

[Liu et al., 2021] Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022.

[Maas et al., 2013] Maas, A. L., Hannun, A. Y., Ng, A. Y., et al. (2013). Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, page 3. Atlanta, GA.

[Polling et al., 2020] Polling, M., Li, C., Cao, L., Verbeek, F., de Weger, L., Belmonte, J., De Linares, C., Willemse, J., de Boer, H., and Gravendeel, B. (2020). Automatic image classification using neural networks increases accuracy for allergenic pollen monitoring.

[Redmon et al., 2016] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788.

[Ren et al., 2015] Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28.

[Rostami et al., 2023] Rostami, M. A., Balmaki, B., Dyer, L. A., Allen, J. M., Sallam, M. F., and Frontalini, F. (2023). Efficient pollen grain classification using pre-trained convolutional neural networks: a comprehensive study. *Journal of Big Data*, 10(1):151.

[Sheikh et al., 2009] Sheikh, A., Panesar, M. S. S., Salvilla, S., and Dhami, S. (2009). Hay fever in adolescents and adults. *BMJ Clinical Evidence*, 2009:0509.

[Tian et al., 2024] Tian, Y., Liu, Y., Lin, B., and Li, P. (2024). Research on marine flexible biological target detection based on improved yolov8 algorithm. *PeerJ Computer Science*, 10:e2271.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

[Wu et al., 2025] Wu, B., Feng, S., Jiang, S., Luo, S., Zhao, X., and Zhao, J. (2025). Eb-yolo: An efficient and lightweight blood cell detector based on the yolo algorithm. *Computers in Biology and Medicine*, 192:110288.

[Zhang et al., 2024] Zhang, C.-J., Liu, T., Wang, J., Zhai, D., Chen, M., Gao, Y., Yu, J., and Wu, H.-Z. (2024). Deeppollencount: a swin-transformer-yolov5-based deep learning method for pollen counting in various plant species. *Aerobiologia*, 40(3):425–436.

[Zuberbier et al., 2014] Zuberbier, T., Lötvall, J., Simoens, S., Subramanian, S., and Church, M. K. (2014). Economic burden of inadequate management of allergic diseases in the european union: a ga2len review. *Allergy*, 69(10):1275–1279.