University of Novi Sad
Faculty of Sciences
Department of Mathematics and Informatics

Simona Fimić

Detection of Raspberries in Images using Deep Learning

Master Thesis

Mentor:

Oskar Marko, PhD

Novi Sad, 2024

I wish to extend my sincere gratitude to PhD student Dimitrije Stefanović for their invaluable guidance, assistance and unwavering belief in the potential of this research. I would also like to thank my esteemed mentor Dr Oskar Marko, whose mentorship and encouragement have ignited my passion for Data Science and Artificial Intelligence. I am also thankful to the faculty members of Department of Mathematics and Informatics for their dedication to education and their role in shaping my academic journey. I would also like to thank my friends for incredible support, their belief in me has kept my spirits and motivation high during this process. Finally, I would like to extend my deepest gratitude to my family for their unwavering love and support. None of this would have been possible without them, and I dedicate this work to my family.

Contents

List of Figures and Tables

1. Introduction

Agriculture, as the cornerstone of food production, plays a key role in sustaining global populations. With the ever-growing demand for food to feed an expanding world population, the agricultural sector faces the challenge of optimizing processes to ensure efficient and sustainable production. In this dynamic landscape, technological advancements, particular in the realm of computer vision and artificial intelligence, have emerged as powerful tools to address the complexities of modern agriculture. The integration of cutting-edge technologies into agricultural practices not only enhances productivity but also introduces novel solutions to age-old challenges. As the agricultural sector embraces precision farming and smart agriculture, the application of deep learning algorithms holds great promise in revolutionizing various aspects of crop management, from monitoring plant health to automating harvesting processes. Deep learning, a subfield of artificial intelligence, has demonstrate remarkable success in tasks related to image recognition and object detection. As the demand for intelligent and automated systems continues to grow, the application of deep learning algorithms to real-world scenarios has become increasingly prevalent. Detection and recognition of specific objects within images play a crucial role in diverse fields, ranging from surveillance and agriculture to robotics. Within this context, this master thesis focuses on the intersection of deep learning and agriculture, with a specific emphasis on the detection of raspberries.

Figure 1: Raspberries (Rubus idaeus)

Rubus is a large, diverse and widely distributed genus that includes approximately 900 to 1000 species [1]. Red Raspberry (Rubus idaeus) (Figure 1) is the most popular raspberry species grown commercially in temperate climates, including the Baltic region. They are widely cultivated fruits known for its vibrant color and distinct shape. The structure of raspberry cultivars and genetic resources in the Baltic countries have been influenced by the historical

political situation in the 20th century and climatic conditions, especially winterhardiness. The genetic resources consist of some old European and American cultivars, but mostly of cultivars and hybrids bred in Russia [2]. The accurate identification and localization of raspberries in images hold significant implications for automated harvesting processes, quality control in agricultural settings, and the overall optimization of raspberry production. Achieving precise and efficient raspberry detection is a complex task, as it involves addressing challenges related to varying lighting conditions, occlusions and the natural diversity in raspberry appearances. In the realm of precision agriculture, the fusion of cutting-edge technologies holds immense promise for optimizing crop management practices. The emergence of sophisticated technologies, particularly in the field of computer vison, has revolutionized the way we approach agricultural challenges. Computer Vision, often abbreviated as CV, is defined as a field of study that seeks to develop techniques to help computers see and understand the content of digital images such as photographs and videos. Object detection, a subdomain of computer vision, represents a technique for locating instances of objects in images and videos and it offers a powerful solution to the problem of raspberry detection [3]. Applied to agriculture, these techniques hold the promise of enabling rapid and accurate identification of raspberries and their classes, reducing the time and resources needed for timely intervention.

Application of detecting raspberries through object detection offers a range of practical and impactful applications across different industries. In precision agriculture, object detection can be employed to monitor raspberry crops. Drones equipped with cameras or ground-based sensors can detect the presence of ripe or diseased raspberries, allowing for targeted interventions such as selective harvesting or disease management [4]. Also, object detection systems can be integrated into food processing lines to inspect batches of raspberries. This ensures that only high-quality fruits meet the desired standards for packaging and distribution, reducing waste and improving overall product quality [4]. Moreover, supermarkets and grocery stores can use object detection to automate the monitoring of raspberry stocks and this not only streamlines inventory management, but also helps in preventing stockouts or overstock situations [5]. Another application can be in environmental monitoring where it can be deployed in natural environments or wild habitats where raspberries grow. This can aid researches in studying the ecological impact of raspberries on local ecosystems or monitoring the health of wild raspberry population [6]. Educational applications or platforms can use object detection to teach students about different types of fruits, including raspberries. This interactive learning experience enhances engagement and knowledge retention [7]. Furthermore, object detection can assist in monitoring for pests that affect raspberry crops. Early detection allows for targeted pest control measures, reducing the need for widespread pesticide application and minimizing environmental impact. By applying object detection to raspberries, these scenarios demonstrate the versatility and broad utility of this technology, showcasing its ability to enhance efficiency sustainability, and user experiences across various domains.

Fresh raspberries have a very short shelf life and are generally only readily available around summer. Most of the produces raspberries worldwide are processed, i.e., frozen and sold within different frozen fraction blocks or in jams and sauces. However, there has been an increasing demand for fresh raspberries out-of-season lately, and so many producers appear to be interested

in growing primocane fruiting raspberry cultivars [8]. Today, raspberry fruit is being used in the production of bakery goods, jams, jellies, beverages, dairy products like ice cream and yogurt, fruit syrups and many other specialty products such as fruited honey. The most significant health benefits of raspberry fruits are attributed to the phenolic compounds, such as flavonoids, phenolic acids, and tannins [9]. It is believed that raspberry has a higher antioxidant capacity than most other fruits and vegetables [10].

In the context of this thesis, the focal point of my research lies withing the domain of detecting raspberries, which are provided by a homogenized dataset of images taken on-site at the Institute of Horticulture in Dobele, Latvia. Raspberries are the third most important berry crop in Latvia by the total area of plantations. Winter-hardiness is the main limiting factor in the growing of this crop. Winter-hardiness is a complex trait consisting of several components and most important of these are the combination of high cold resistance with the ability to regain hardiness after long thaw periods, especially in the second half of the winter, when deep dormancy of raspberries has ended (in December) [11].

In Latvia the main part of commercial raspberry cultivars are varieties of Russian breeding. Many of these have high cold hardiness, but worse restorative ability after thaws that are common in the Latvian climate. Besides, the majority of Russian cultivars lack berry quality - they are too soft and have short storage when transported and sold. The widely grown Western European cultivars ‘Glen Ample’ and ‘Tulameen’, which are characterized by high berry quality, have low winter-hardiness in Latvia [11]. Leading suppliers of raspberries are Netherlands, Poland and Germany with a combined 79% share of total imports, while leading importers from Latvia are Estonia and Lithuania. Moreover, the average export price for raspberries in Latvia in 2022 were $10,711 per ton, waning by -22.7% against the previous year and import price stood at $11,469 per ton, falling by -11.2% against the previous year [12].

The world production of raspberries from 2010 to 2018 is increasing every year, with the demand itself growing and therefore is a need for increasing production. Although in this period the world production has grown by about 300 000 tons, every year there is a growing demand for the offered available quantity on the market [13].

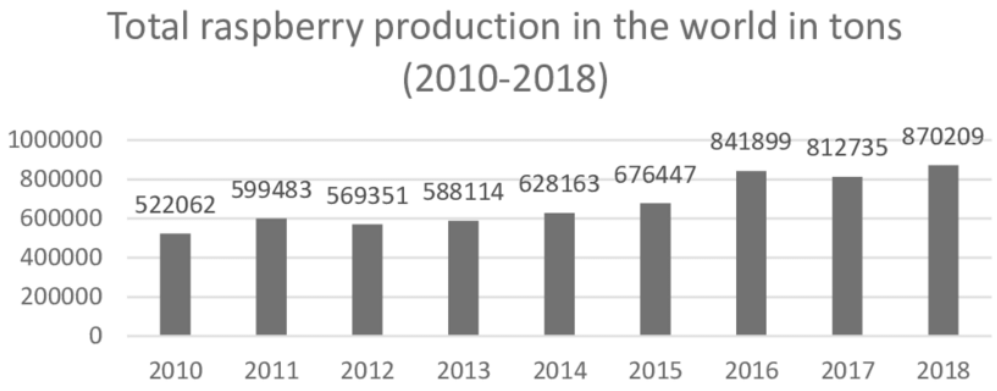

Figure 2: Raspberry production in the World (quantity in tons)

In this period, the largest world raspberry production is evidenced in 2018 with 870 209 tons, while the lowest production is in 2010 with a total world production of 522 062 tons (Figure 2).

Demand for raspberries and blackberries has risen sharply in Europe and North America in recent years. [4] In the Western countries, raspberries are considered a luxury product. The data in Figure 3 show that Russian Federation with a production of 165 800 tons of raspberries in 2018 occupies 19% of world production and is in first place in the world in terms of raspberry production, Mexico is in second place with total production of 130 187 tons and the third largest raspberry producer is Serbia with total production of raspberries of 127 010 tons in 2018 [13]. Nearly one quarter of the world's total raspberry production is generated in Serbia, with almost all of the production being exported [14].

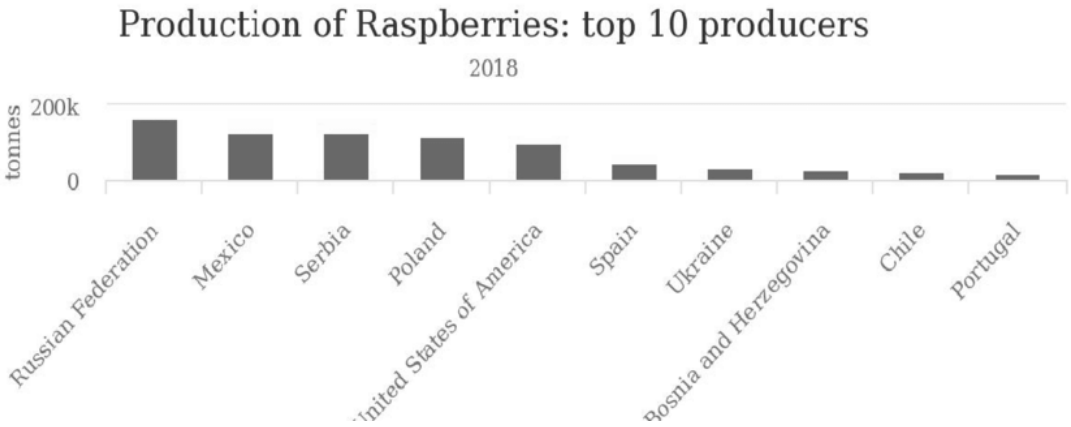

Figure 3: Production of Raspberries: top 10 World produced in tons

Also, from 2018 to 2023 there were not any big changes, because Russia, Mexico and Serbia remained the top countries with the highest volumes of production with a combined 47% share of global production [13].

In this thesis, we delve into the field of raspberry detection using state-of-the-art object detection algorithms. Through a meticulous exploration of deep learning techniques and innovative methodologies, we seek to harness the potential of computer vision to address the unique challenged posed by Raspberry detection. By doing so, we aspire to contribute to the advancement of sustainable and efficient agricultural practices, safeguarding crop yields and promoting ecological equilibrium. This thesis focuses on leveraging the potential of object detection algorithms, YOLO (You Only Look Once), Detectron2 and YOLONAS to automatically identify, localize, predict and analyze Raspberry within agricultural imagery.

This thesis is organized into 6 chapters, where the first and 6th chapter provide an introduction and a conclusion. Chapter 2 makes a review of a topic, going through the literature that solves the same or similar problems. Methodologies, models, data processing and algorithms used in this thesis are described in Chapter 3. Each combination of results is presented and discussed in detail in Chapter 4. Also, Chapter 5 provides some Additional Analyses.

2. Related Work

Deep neural networks (DNN) are widely used in applications that are related to agriculture. Different deep learning algorithms can be more or less successfully trained to process, annotate and detect various fruits, or specially in raspberry behaviors. For example, DNN model based on YOLOv5 architecture for Quince and the same raspberry dataset that I used in this master thesis on RGB images was used in [15]. YOLOv5 in this research provides sufficiently good performance and precision trade-off over 7 berry classes that are related to the berry development stage. It is useful in the process of quince and raspberry phenotyping for the agriculture experts, where the yield and berry size parameters have to be estimated. Using their DNN model, they have showed that it is possible to achieve a mean Average Precision close to 80.9% and in some cases (Average Precision) close to 95% for some classes.

The research study [16], from which we obtained the original raspberry dataset, notably did not employ any specific model for training and detection purposes. Instead, it exclusively furnished a meticulously annotated dataset in YOLO format, facilitating subsequent model development and evaluation.

On the other hand, YOLO can be used for Crop Disease Detection [17]. The deep learning model based on YOLOv3 was trained using PlaintVillage dataset that is available on Kaggle. This pre-trained darknet-53 convolutional weights were used for the training and this model can be used for automatic disease detection.

Another paper had processed research about automated weed detection with the help of convolution neural networks, where Python and Tensorflow framework were used to implement the approach [18]. The images captured using the phone and camera were annotated and fed into the neural network model, where accuracy of 98% was achieved using the transfer approach. Moreover, another paper used Convolutional Neural Network-based method to identify corn diseases [19]. In this research, the images were pre-processed to remove the noise and the unwanted details in the sample images and the trained deep convolutional model detected various diseases associated with the corn plants. Models YOLOv3 and YOLOv5 were shown to be able to detect Apples in the Orchards [20] with high accuracy without further complications. To accomplish that, and improve the accuracy of detecting the Apples on a challenging background or environment, the images were pre-processed by thickening the borders, introducing slight blurs and adaptive histogram alignment. Furthermore, in [21] in summary of some works related to studies on fruit-based quality detection was noted that CNNs predominate in DL approaches. Also, in detecting specifically raspberries, they got performance detection rate 91.20% for Faster R-CNN, 84.50% for Statistical Modeling and ANN and 81.00% for Descriptive statistical and ANN.

Another research [22] trained multiple TensorFlow Lite Object Detection Models on an image dataset of 750 pictures of US coins. SSD-MobileNet-FPNLite-320x320 Model has high accuracy (even when trained on a small dataset) with 84.81%, while still running fast enough to achieve near real-time performance, so it is useful for proof-of-concept prototypes. SSD-MobileNet-v2 has sufficient speed while maintaining relatively good accuracy, with 75.33%. If speed is the

most crucial aspect of object detection task, or it can get away with missing a few detections, then this is a good model. The lower accuracy can be improved by using a larger training dataset. EfficientDet-Lite-D0 has comparable speed to SSD-MobileNEt-FPNLite-320x320, but it has worse accuracy on this dataset, with 60.79%. This model can be considered as an alternative if satisfactory accuracy is not achieved with the SSD-MobileNet Models, as its distinct architecture may yield improved performance on specific dataset.

EfficientDet-D0 is slower than other models and performed inaccurately on this dataset with 46.91%, where the poor accuracy is likely due to the limited size of the coin dataset that was used for training. SSD-MobileNet-v1(quantized) Model is the fastest, but also inaccurate with 49.77%. This Model comes from the TensorFlow 1 model zoo and is pre-configured for quantization-aware training, so it theoretically should have better accuracy than other models when quantized.

The research study [23] presented a comprehensive overview and review of a fruit detection and recognition based on DL for automatic harvesting from 2018. Figure 4 displays the distribution of articles per year, network used and crops detected.

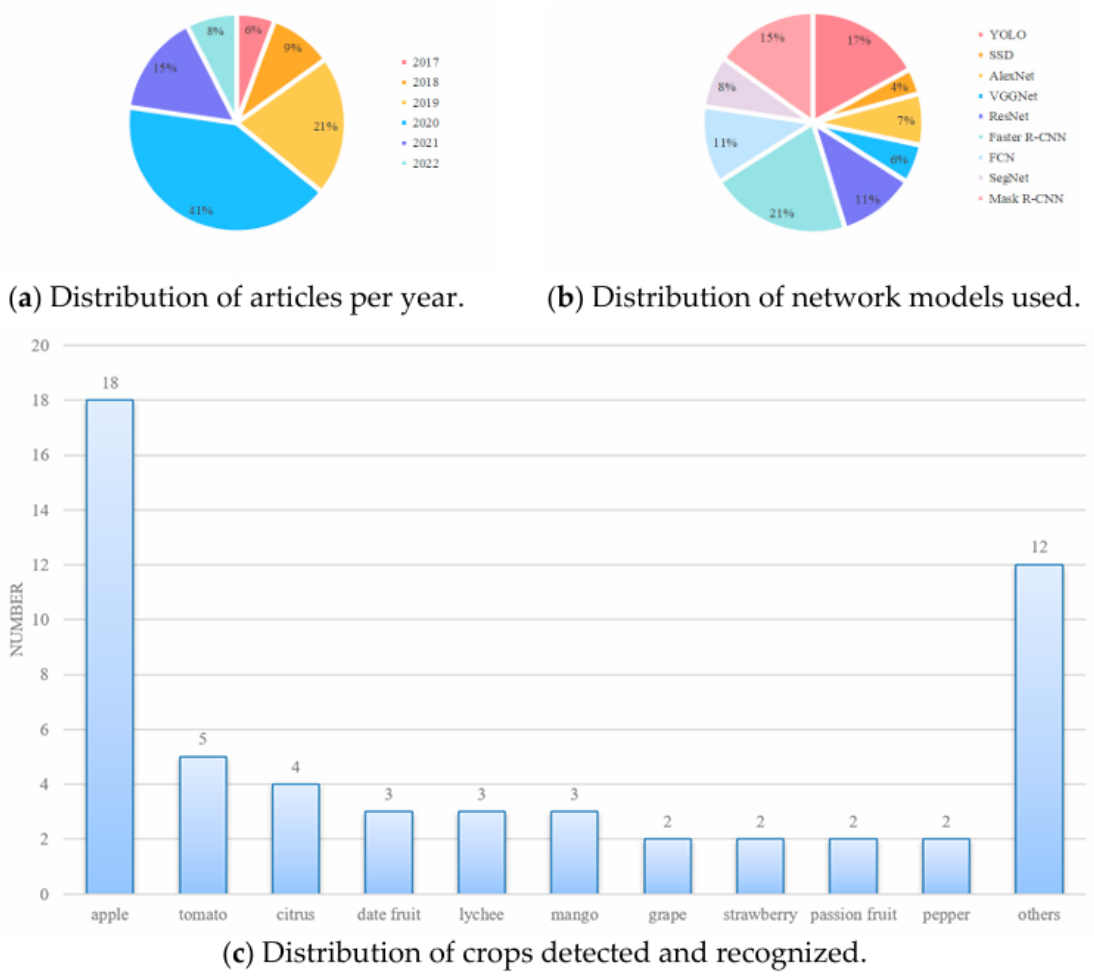

Figure 4: Distribution of articles per year, network used and crops detected

As shown in Figure 4, in recent years the application of DL techniques and robotic systems to automate agricultural processes has garnered significant interest. Improvement and application research based on Faster R-CNN (21%) is currently a hotspot. The recognition accuracy of fruit detection methods based on Faster R-CNN is high, but recognition speed is limited by complex anchor frame mechanisms. When there are mobile deployment and high recognition and speed requirements, fruit detection methods based on YOLO (17%) are used most frequently. In addition, ResNet (11%) is the most popular backbone network, followed by AlexNet.

Most of the research focuses on Apples (32.14%), followed by Tomatoes (8.93%), and Citrus (7.14%). These three kinds of fruits are in high demand and yield globally. There are some reasons that make them ideal candidates for automatic harvesting. Firstly, they individually hang from plants, making them easily detectable based on their distinctive features. Secondly, they have no extreme variations in size or weight. Lastly, they are relatively hard and not easily damaged in mechanical operations. However, in terms of fruit dimensions and peduncle length, different cultivars may exhibit different characteristics, which can affect fruit detection and recognition performance.

In summary, detection of raspberries stands at the intersection of object detection, computer vision and agricultural practices. The synthesis of existing approaches and the exploration of innovative methodologies, such as YOLO-based object detection, Faster R-CNN Model and MobileNet-SSD Models offers a promising avenue to tackle the persistent issues surrounding raspberry crop management and propel the realm of precision agriculture forward. This thesis endeavors to expand upon the groundwork laid by previous studies, aiming to pioneer precise and effective detection methods customized for the distinctive demands of Raspberry cultivation.

3. Materials and Methods

3.1. Data

Dataset used in this thesis consist of data collected from Institute of Horticulture in Dobele in Latvia, where the images were also captured [24][25]. As we mention before, dataset contains annotated images of the raspberries (Rubus idaeus) which were taken in various weather conditions, at different times of the day and from different angles. The images were captured in the four development stages. The 2039 images in the dataset have a resolution of 1773x1773 pixels and were taken by iPhone XS. The annotation was carried out with the help of Label Studio software (version 1.7.1) manually by the experts from Institute of Horticulture in Dobele, and the annotations are presented in YOLO (You Only Look Once) format using ground truth ROI. The YOLO format is a specific format for annotating object bounding boxes in images for object detection tasks. In this format, each image in the dataset should have a corresponding text file with the same name as the image, containing the bounding box annotations for that image. The dataset has five classes, which are:

1. Buds

2. Flowers

3.Unripe Berries

4. Ripe Berries

5. Damaged Buds

Out of 46 659 annotations presented in the dataset, 11 788 was for Buds, 4748 for Flowers, 29 156 for Unripe Berries, 463 for Ripe Berries and 504 for Damaged Buds. The Raspberry images in the dataset were taken in an orchard at the Institute of Horticulture in Dobele in Latvia. The orchard was planted (coordinates: 56°36′23.5″ N, 23°18′09.8″ E) with 14 genotypes of Raspberry. The images were taken at five different stages of the Raspberry's phenological development, which later represents our classes. The distance between the raspberry bushes and the camera was about 30cm, capturing close-up views so that the crop elements could be seen as clearly as possible in the images. The images were taken from a variety of angles; it the lengths of the Raspberry shoots were 1.0-1.4m, then the photographing angle to the soil was 45°. If the lengths of the shoots were around 1.5-1.6m, when the angle was 90°, but if the shoots were longer than 1.7m, then the angle was 120°.

The images were taken under a variety of weather conditions, including sunny, cloudy and partly cloudy. Florican raspberry buds, flowers and unripe berry images were captured from 15 to 16 June 2021, and buds, damaged buds, flowers and unripe and ripe berries were captured on 2 July 2021. Primocane raspberry buds, damaged buds, flowers and unripe and ripe berries were captured on 6 August 2021. Temperature is one of the factors that influence yield, but when plants are grown under uncontrolled conditions (in the open field), it varies from year to year and thus affects the yield elements. For example, low temperature during flowering can affect berry

formation as the flowers are less likely to pollinate. In spring, high temperatures and insufficient moisture supply can intensify winter damage, resulting in bud dieback or the death of corroding shoots, which affects the overall view of yield elements. This may be less important for the identification of the object themselves, but it certainly has an impact on yield and berry size.

The dataset uploaded to the Institute of Electronics and Computer Science (EDI) consists of raw images of red raspberry fruit, each saved in the .jpg format. The dataset includes .txt files in the YOLO format, which provide annotations for the locations of the raspberry fruit in the images using bounding boxes. As I mention before, these annotations were created using the Label Studio software and may overlap to cover the entire berry. The YOLO format stores the annotations in .txt files: 0-buds, 1-flowers, 2-unripe berries, 3-ripe berries, and 4-damaged buds, and the following values indicate the x and y coordinates, as well as the height and width of the bounding box. (Figure 5)

Figure 5: Detection results obtained with the trained detector

The process of raspberry breeding takes 15-20 years from crossing to cultivar. To select candidates for cultivars, the characteristics of several thousand seedlings must be described and evaluated, most of which is performed visually. This is a time-consuming and labor-intensive process that also requires sufficient manpower. In addition, visual scoring is relatively subjective, and results may vary among different evaluators. Therefore, the utility of new techniques for non-invasive fruit detection and phenotyping to improve yield performance should be evaluated by adopting Machine Learning (ML) techniques, considering cost-benefit and human-centered considerations. ML and Deep Learning (DL) techniques have shown very promising results in fruit classification and detection problems [26] and yield quality evaluation [27]. A neat and clean image dataset in precision agriculture supplemented with an image labeling tool is the basic requirement to build accurate and robust ML models for the real-time environment. The annotated raspberry rubus idaeus dataset is a comprehensive collection of images and annotations of the fruit, specifically designed for use in the field of DL.

3.2. Data Processing

As we mention before, annotations from these images were carried out manually by the experts from Institute of Horticulture in Dobele in Label Studio software, so we had already annotations that we could immediately use. In the Data Processing phase, a series of image manipulation techniques were applied to preprocess the dataset to improve the performance of the detection models. These preprocessing steps were crucial for enhancing the quality of the images and ensuring they were suitable for training Deep Learning models. To correct any orientation discrepancies in the images, we used and Auto-Orient preprocessing technique. This technique ensured that all images were aligned consistently, which is important for accurate object detection. Next preprocessing step was resizing images.

The images were resized to dimensions compatible with the input requirements of the various deep learning models utilized in this study (mainly they were resized into format 640x640). Resizing helps to standardize the input size, reducing computational complexity and ensuring uniformity across the dataset.

The next step for improvement and fixing misclassification was function Color Conversion which applies different color space transformations to enhance the visual distinction between different classes. One of the advanced preprocessing techniques is Normalization which standardizes the pixel values to a specific range, which helps in stabilizing the training process and improving convergence. One of the most important preprocessing techniques is Noise Reduction. Techniques such as Gaussian blur were used to reduce image noise, making the features more distinguishable. Gaussian blur uses two mathematical functions (one for the x-axis and one for the y-axis) to create a third function, also known as a convolution. This third function creates a normal distribution of those pixel values, smoothing out some of the randomness. The equation of a Gaussian function in one dimension is:

$$G_{xy} = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$

In two dimensions, it is the product of two such Gaussians, one per direction:

$$G_{xy} = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Where x is the distance from the origin in the horizontal axis, y is the distance from the origin in the vertical axis, and σ is the standard deviation of the Gaussian distribution. When applied in two dimensions, this formula produces a surface whose contours are concentric circles with a Gaussian distribution from the center point. Values from this distribution are used to build a convolution matrix which is applied to the original image. Each pixel's new value is set to a weighted average of that pixel's neighborhood. The original pixel's value receives the heaviest weight (having the highest Gaussian value) and neighboring pixels receive smaller weights as their distance to the original pixel increases. This results in a blur that preserves boundaries and edges better than other, more uniform blurring filters. Other techniques such as Offset, Dark and

Flat Corrections were applied to adjust for any systematic errors in the image acquisition process, such as varying lighting conditions or sensor artifacts.

The reason why damaged buds class performed poorly is its limited representation in the dataset, with only 504 annotations out of 46 657 total annotations in the dataset .Consequently, it was decided to remove the damaged buds class from the dataset to focus on improving the detection accuracy for the remaining classes. By refining the dataset through these preprocessing techniques and strategically removing problematic classes, the quality of the input data was significantly improved, leading to better training outcomes and more accurate detection results in the subsequent experiments.

3.3. Algorithms

Object detection aims to identify the location, size, spatial relationship and classes of specific targets in images or video sequences. It always had high research and application value in face recognition, industrial defect detection, UAV aviation detection, traffic and vehicle detection and other fields. In recent years, with the update and iteration of high-performance GPUs, the continuous emergence and improvement of various large-scale datasets (such as DOTA, ImageNet, COCO) and the strong rise of neural networks based on convolutional object detection algorithms using deep learning techniques have become the mainstream of contemporary object detection technology.

In general, deep learning-based object detection models consist of a backbone and a head network, where the basic structures of deep learning-based object detection models are described in Figure 6 [28].

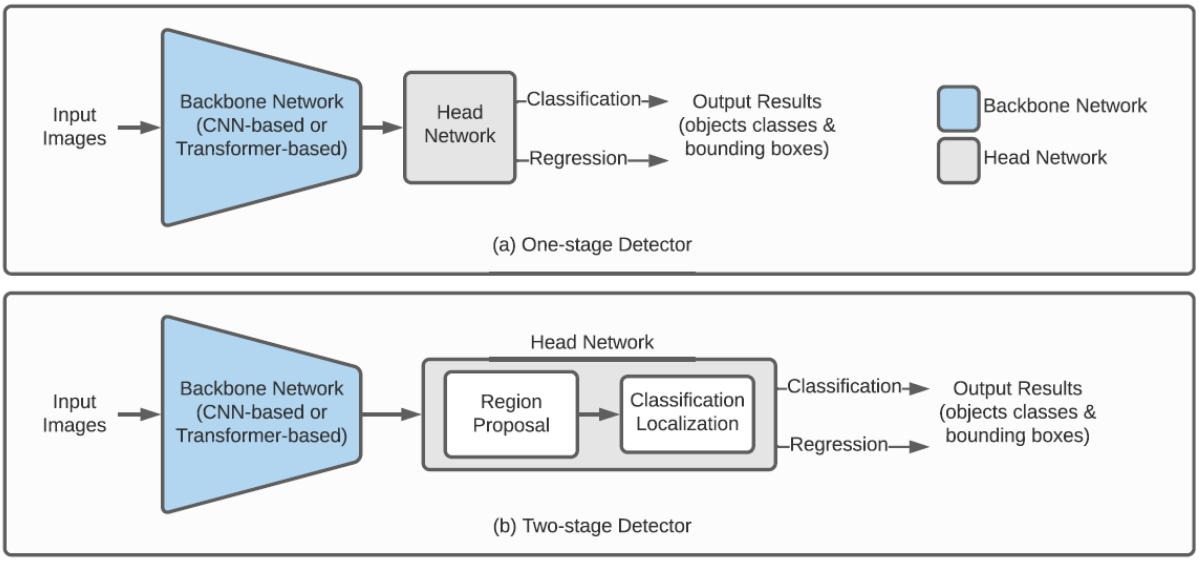

Figure 6: Basic deep learning-based one-stage vs. two-stage object detection model architectures. The backbone network can be used as a CNN or transformer-based network, where the backbone network can be categorized into one or two stage networks according to the structure of the head network. As shown in a) One-stage detector simlutaneously object localization and classification in the head network. On the other hand, localization and classification in the two-stage detector are performed on regions after the region proposals are obtained, as shown in b).

The backbone network is capable of extracting features from the input images, while the head network uses the extracted features to localize the bounding boxes of the detected objects and classify them. In the case of backbone networks, CNN-based networks are commonly employed. [28] On the other hand, the head network is divided into one-stage or two-stage detectors.

While the one-stage detector performs object localization and classification simultaneously, the two-stage detector performs classification to determine classes after proposing the regions of interests (ROIs).

One-stage methods prioritize inference speed and example models include YOLO [29], RetinaNet [30], SSD [31], MobileNet [32] and EfficientDet [33]. In the one-stage method, the detection process is simplified because this method directly predicts bounding boxes and class probabilities for objects in a single step simultaneously. It does this by first extracting features using a CNN, the image is divided into a grid of squares, where for each grid cell, the model predicts bounding box and class probabilities. They are designed to be faster and more efficient, often making them suitable for real-time applications, but they are often less accurate than two-stage methods, particularly for small or overlapping objects.

Two-stage methods prioritize detection accuracy, and example models include Faster R-CNN [34], Mask R-CNN [35], Detectron2 [36], SPP-Net [37] and Cascade R-CNN [38]. Two-stage methods first generate region proposals (candidate bounding boxes) and then classify those proposals into object categories while refining their bounding boxes. This two-step process generally results in higher accuracy, but can be slower. In essence, one-stage methods prioritize speed and efficiency, making them suitable for applications where real-time processing is essential. In contrast, two-stage methods focus on accuracy and robustness, often at the expense of speed, making them better suited for scenarios where precision is critical. The choice between these methods often depends on the specific requirements of the application at hand.

In this master thesis, we utilized several state-of-art models for object detection, including two one-stage methods YOLOv5, YOLONAS and one two-stage method, Detectron2. Each of these models was chosen for their unique strengths and capabilities in accurately detecting and classifying objects within the dataset.

3.3.1. YOLOv5 Model

YOLO is a state of the art, real-time object detection algorithm created by Joseph Redmon, Santosh Divvala, Ross Girshick and Ali Farhadi in 2015 and was pre-trained on the COCO dataset [39]. It uses a single neural network to process an entire image.

The image is divided into regions and the algorithm predicts probabilities and bounding boxes for each region. It is an object detection and localization network that streamlines the detection process by simultaneously predicting object classes and bounding box coordinates in a single pass, making it ideal for real-time applications.

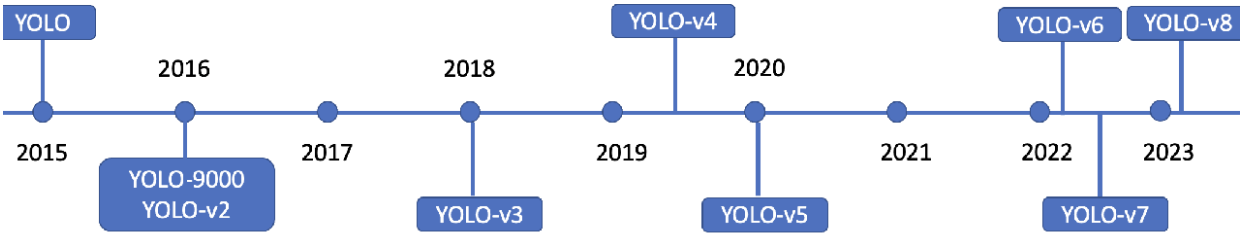

Figure 7: Timeline of You Only Look Once (YOLO) variants

There are also different variants of YOLO methods as shown in Figure 7 [40]. YOLOv1[41] was introduced in 2016, while YOLOv2 (or YOLO9000) [42] was realized in 2017 and used a Darknet-19 network architecture and incorporated batch normalization and anchor boxes. It also introduced multiscale predictions, enabling the detection of objects at different scales. Then, in 2018 YOLOv3 [43] was released and used a Darknet-53 network architecture, introduced feature pyramid networks (FPNs) and used residual connections. It also used anchor boxes with dynamic scaling, leading to better detection at different scales.

Next YOLO version, YOLOv4 [44] was released in 2020, which used a CSPDarknet-53 network architecture, added various improvements such as spatial pyramid pooling, and used a complex data augmentation pipeline. Additionally, YOLOv4 also included efficient training approaches such as mosaic data augmentation and focal loss. YOLOv5 [45] was a PyTorch implementation rather than a fork from the original Darknet, which has a CSP backbone and PANET neck. The major improvement included auto-learning bounding box anchors.

YOLOv6 [46] and YOLOv7 [47] were released in 2022, where YOLOv6 was a single-stage object detection framework dedicated to industrial applications, with hardware-friendly efficient design and high performance, while YOLOv7 reduced the number of parameters by 75%, required 36% less computation and achieved 1.5% higher AP (average precision) compared with YOLOv4.

Finally, YOLOv8 model was released in 2023 which is fast, can be applied to a wide range of object detection and image segmentation tasks and can be trained on large dataset and run-on various hardware platforms, from CPUs to GPUs.

YOLOv5 continues the consistent idea of the YOLO series in algorithm design: namely, the image to be detected was processed through a input layer (input) and sent to the backbone for feature extraction [48].

The backbone obtains feature maps of different sizes and then fuses these features through the feature fusion network (neck) to finally generate P3, P4 and P5 (in the YOLOv5, the dimensions are expressed with the size of 80x80, 40x40 and 20x20) to detect small, medium and large objects in the picture, respectively. After the three feature maps were sent to the prediction head (head), the confidence calculation and bounding-box regression were executed for each pixel in the feature map using the preset prior anchor, so as to obtain a multidimensional array (BBoxes) including object class, class confidence, box coordinates, width and height information. By setting the corresponding thresholds (confttthreshold, objthreshold) to filter the useless information in the array and performing anon-maximum suppression (NMS [49]) process, the final detection information can be output.

The process of converting the input picture into BBoxes is called the Inference process and the subsequent threshold and NMS operations are called post-processing. The post-processing does not involve the network structure. The default inference process of YOLOv5 can be represented by Figure 8 [48].

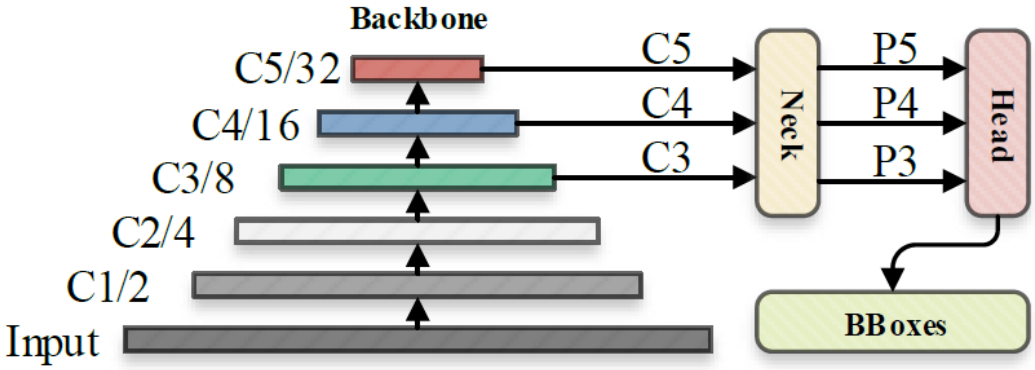

Figure 8: The default inference flowchart of YOLOv5

In the details of bounding-box regression, different from the previous version of YOLO algorithm, the process of YOLOv5 can be explained by the next equations (1):

$$\begin{aligned} g_x &= 2\sigma(s_x) - 0.5 + r_x \\ g_y &= 2\sigma(s_y) - 0.5 + r_y \\ g_h &= p_h(2\sigma(s_h))^2 \\ g_w &= p_w(2\sigma(s_w))^2 \end{aligned} \quad (1)$$

In the above formula, set the coordinate value of the upper left corner of the feature map to (0,0). Then, r_x and r_y are unadjusted coordinates of the predicted center point. Next, g_x, g_y, g_w, g_h represents the information of the adjusted prediction box. Also, p_w and p_h are for the

information of the prior anchor. Finally, s_x and s_y represents the offset calculated by the model. The process of adjusting the center coordinate and size of the preset prior anchor to the center coordinate and size of the final prediction box is called bounding-box regression.

The main structure of a backbone of YOLOv5 is the stacking of multiple CBS (Conv + BatchNorm + SiLU) which represents a sequence of three operations often used in neural networks, particularly CNN, to process data. Layer Convolution applies convolutional filters to the input data to extract spatial features and it involves sliding a filter or kernel over the input to produce feature maps.

Then, Batch Normalization normalizes the output of the previous layer in sense that it scales and shifts the data to have a mean of zero and a variance of one, which helps stabilize and speed up training by reducing internal covariate shift. Sigmoid Linear Unit (SiLU) also known as the Swish activation function is defined in (2), where $\sigma(x)$ is a sigmoid function which is shown in Figure 9 [50].

$$SiLU\ (x) = \ x\sigma(x) = x(\frac{1}{1+e^{-x}}) \tag{2}$$

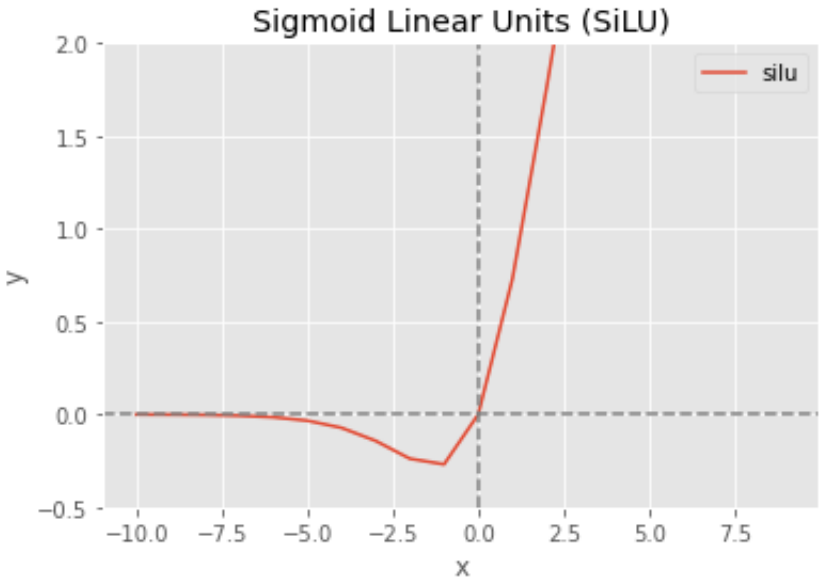

Figure 9: Sigmoid Linear Units (SiLU)

This activation function, $\sigma(x)$ helps introduce non-linearity and has been shown to perform well in various deep learning models. Combining these three components, a CBS block can help improve the learning capability and performance of a neural network by efficiently extracting features, normalizing activations and introducing non-linearity [51].

As we mention before, the main structure of a backbone of YOLOv5 consists multiple CBS models and C3 modules and finally one SPPF module is connected. CBS module is used to assist C3 module in feature extraction, while SPPPF module enhances the feature expression ability of the backbone. The backbone of YOLOv5 is shown in Figure 10 [48].

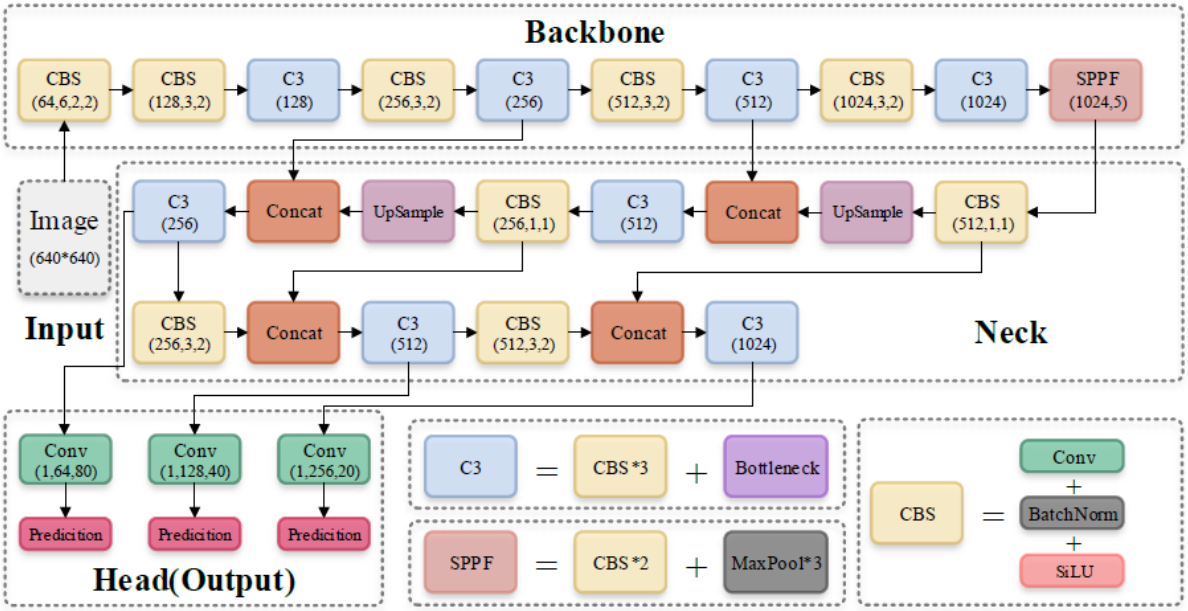

Figure 10: Default network structure of YOLOv5

Therefore, in the backbone of YOLOv5, the most important layer is the C3 module, which is represented in Figure 11 a). [52]

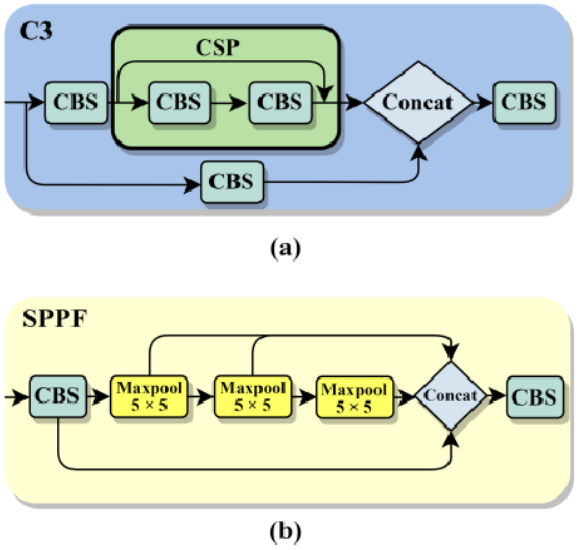

Figure 11: Structure of YOLOv5: a) C3 module, b) Spatial pyramid pooling fast module

The basic idea of C3 comes from CSPNet (Cross Stage Partial Networks [53]). CSPNet was introduced to enhance the learning capability of convolutional neural networks by partitioning

feature maps into two parts and merging them through a cross-stage hierarchy. The C3 module consists of three conv- and cross-stage partial (CSP) structures and residual connections, which can enhance feature extraction ability while ensuring the model is lightweight. Another module inside the backbone of YOLOv5 is SPPF, which is represented in Figure 11 b) [52]. The spatial pyramid pooling fast (SPPF) module consists of a CBS module and several max pooling layers, which fuse the feature maps of different receptive fields and enrich the expression ability of feature maps.
In the neck, YOLOv5 uses the methods of FPN and PAN, as shown in Figure 12 [48]. The deep layers in the backbone extracts high-level features such as textures or parts of objects, whereas the shallow layers extract low-level features such as edges and curves. The Feature Pyramid Network (FPN) uses both high and low-level features effectively to hierarchically stack feature maps through top-down pathway with lateral connection to detect objects of various sizes [54]. Through this method, the performance of detecting multi-scale objects was improved and computational cost and memory problems were solved.

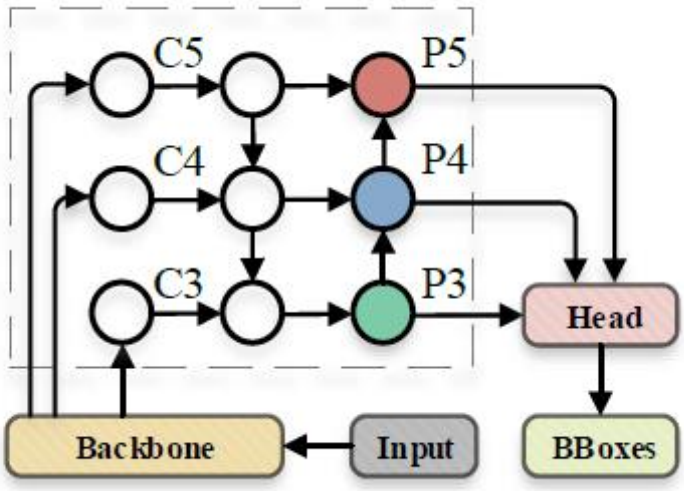

Figure 12: The dotted line in the figure is the default feature fusion path of YOLOv5

The basic idea of FPN is to up-sampling the output map (C3, C4 and C5) generated by multiple new feature maps (P3, P4 and P5) for detecting different scales targets. Moreover, the feature fusion path of FPN is top-down [48]. On this basis, PAN reintroduces a new bottom-up feature fusion path, which further enhance the detection accuracy for different scales objects.

Finally, the head in YOLOv5 is responsible for generating the final object detection predictions from the processed feature maps. It outputs bounding box coordinates, class probabilities and object confidence score for each detected object, allowing YOLOv5 to identify and classify objects in the input image efficiently [55]. The multi-scale prediction approach enhances the model's ability to detect objects of varying sizes.

3.3.2. Detectron2 Model

Detectron2 is a state-of-the-art open-source library developed by Facebook AI Research (FAIR) for computer vision tasks. It is built on top of the PyTorch deep learning framework and is used for object detection, segmentation (both instance and semantic), and other related tasks like keypoint detection and panoptic segmentation. It is the successor of Detectron and the MaskRCNN Benchmark [56] and it is considered to provide state-of-the-art results in both types of tasks. It implements: Mask R-CNN, RetinaNet, Faster R-CNN, TensorMask, DensePose and other object detection methods [57]. In terms of segmentation, three different types are supported: (1) Semantic segmentation, (2) Instance segmentation and (3) Panoptic segmentation. Semantic segmentation is an approach that provides a pixel-based classification of shapes. However, it does not distinguish different samples within the same category. Therefore, an instance segmentation approach is used to obtain the object's unique structure [58]. The last object detection segmentation method, Panoptic segmentation tries to create a unifying framework between instance and semantic segmentation [59].

Originally, Detectron2 was trained using the images that contain objects from daily life, not abstract shapes such as M-A islands. However, Detectron2 allows transfer learning i.e., further training an object detection model quickly with a custom dataset from a different domain. Considering Base R-CNN with Feature Pyramid Network (FPN) as an example, the architecture of Detectron3 mainly consists of three parts: backbone network, region proposal network (RPN) and ROI head (box head) as shown in Figure 13 [60]. The backbone network provides feature maps (P1-P5) to the region proposal network (RPN). The ROI head locates (bbox) and segments (mask) objects, together with the corresponding class.

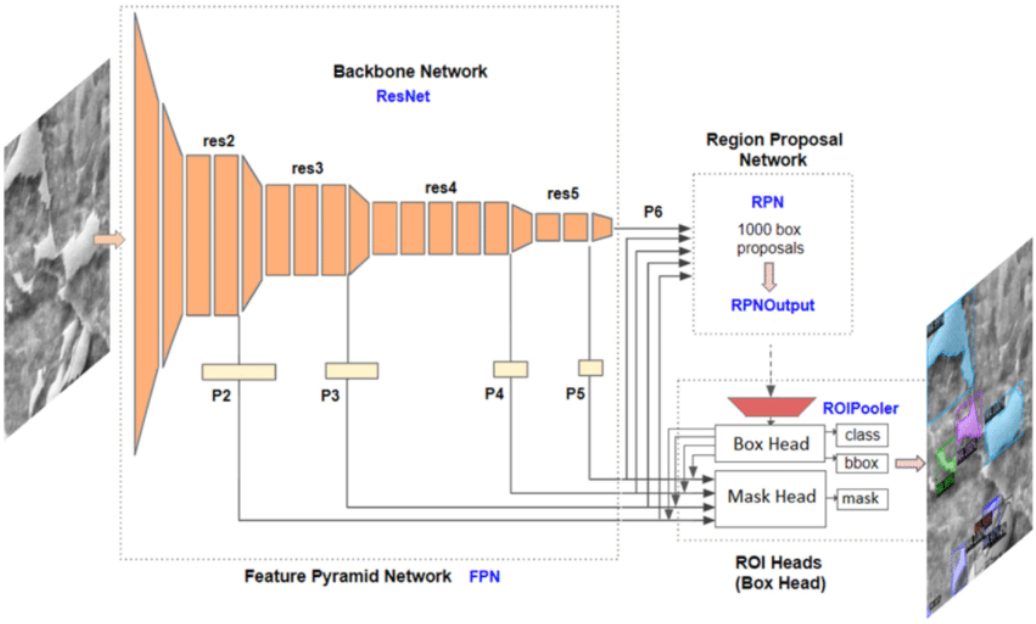

Figure 13: Schematic architecture of Detectron2

From the input image, the backbone network can extract multi-scale feature maps with varying receptive fields. The RPN can detect object regions from the multi-scale feature maps and by default, it will output 1000 proposed boxes with a confidence score. Finally, the box head can crop the feature maps into different sized features and find the box locations and classification labels through fully connected layers. In other words, models like Faster R-CNN in Detectron2 start by generating a set of candidate regions (Region Proposals) that might contain objects and these regions are then classified as containing an object or not (background).

The model also refines the bounding boxes around detected objects to better fit them, further distinguishing them from the background [61]. After detection, Non-Maximum Suppression (NMS) is applied to remove redundant overlapping boxes, helping to isolate the detected objects from the background. During training, the model learns to differentiate between objects of interest and the background. This is crucial because, during inference, the model must decide whether a detected region contains an object or just background.

While Detectron2's object detection models don't perform explicit background segmentation like segmentation models, they do involve background segregation as part of the process of identifying and localizing objects within an image.

Before training, the model requires configuration based on the custom dataset. Also, the Detectron2 repository offers a large number of pretrained models for object detection. Furthermore, one of the main steps in training is to configure the model with the right hyperparameters. Some of the key modifications are custom trainer, augmentation and hyperparameter tuning-to allow training on the dataset. Detectron2 allows developers to build their own custom trainer with its own training logic.

During training, it is of interest how well the model performs on the validation set to see if there is a risk of overfitting the model to the training data. This is considered by setting the hyperparameter cfg.TEST.EVAL_PERIOD which represents iterations after which to evaluate the model. However, by default the COCOEvaluator only evaluates the Average Precision (AP) metric. Moreover, Image Augmentation is necessary to increase the performance of deep networks when building a powerful image classifier with limited training data. This technique artificially creates training images through combination of multiple processing steps, such as random rotation, shifts, shear and flips, etc.

Multiple hyperparameters influence how the model is fitted to the data during the training phase, as well as the performance of the resulting machine learning models. In order to tune the hyperparameters such as learning rate (cfg.SOLVER.BASE_LR), the open-source hyperparameter optimization framework Optuna was used [62].

Detectron2 stands out as a robust and versatile library for computer vision, particularly in the realms of object detection and segmentation. Its architecture, built on the strength of PyTorch, facilitates a range of advances methodologies, including Mask R-CNN, RetinaNet and panoptic segmentation, catering to diverse application needs. The library's ability to support transfer learning and customization through various hyperparameters makes it adaptable to unique datasets and domains, enabling practitioners to harness its full potential effectively.

With a wealth of pretrained models and an emphasis on techniques like image augmentation and hyperparameter tuning, Detectron2 not only enhances the performance of deep learning but also provides a comprehensive framework for experimentation and innovation in computer vision tasks.

3.3.3. YOLONAS Model

YOLONAS stands as an advanced foundational model for object detection, inspired by YOLOv6 and YOLOv8. This model was released in May 2023 by Deci, a company that develops production-grade models and tools to build, optimize and deploy deep learning models [63]. YOLONAS is designed to detect small object, improve localization accuracy and enhance the performance-per-compute ratio, making it suitable for real-time edge-device applications. It introduces a new quantization-friendly basic block, advanced training schemes, post-training quantization, AutoNac optimization and pre-training on top datasets.

The novelty of YOLONAS includes quantization-aware modules, called QSP and QCI that combine re-parameterization for 8-bit quantization to minimize the accuracy loss during post-training quantization. Furthermore, its automatic architecture design uses AutoNac, Deci's proprietary NAS technology. Also, YOLONAS includes hybrid quantization method to selectively quantize certain parts of a model to balance latency and accuracy instead of standard quantization, where all the layers are affected. YOLONAS introduces new architectural innovations through Neural Architecture Search (NAS). NAS involves automatically searching for the best neural network architectures for a given task, aiming to improve efficiency and effectiveness beyond manual design.

It is a technique used in the field of deep learning and artificial intelligence that aims to mechanize the process of simulating neural network designs [64]. The objective of NAS is to discover neural network topologies that are effectively customized for a specific task or dataset, to do so while minimizing the requirement for labor-intensive human design and hyperparameter tweaking.

Using NAS technology, the laborious process of constructing deep learning models may be streamlined and automated and the technology can also be used to construct deep neural networks quickly and efficiently, with the ability to adapt these networks to fit specific production requirements [65].

The overall structure of the NAS is depicted in Figure 14 [64].

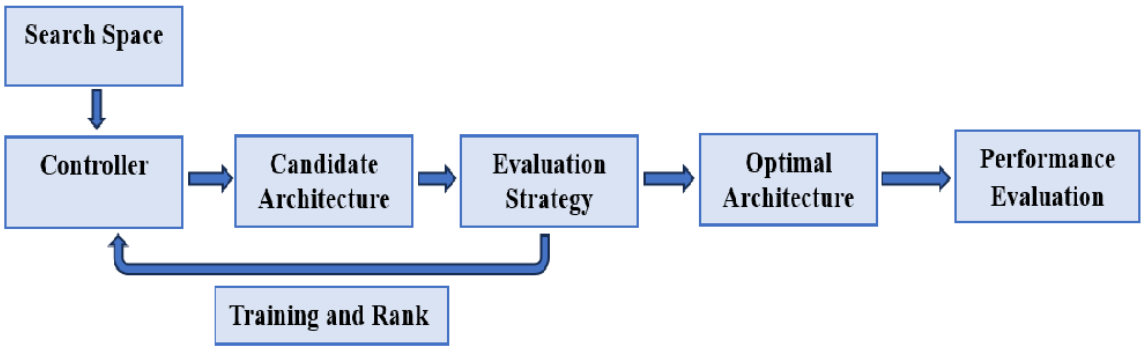

Figure 14: The architecture of NAS general framework

YOLONAS typically follows a neural architecture search pipeline, which involves searching for the best neural network architecture to maximize both accuracy and efficiency. The architecture search process may include the following steps:

(1) Search Space Definition: Define a search space of neural architectures, including choices for various architectural elements such as the number of layers, layer types, kernel sizes and other hyperparameters. Moreover, this search space can be quite extensive [66].

(2) Objective Function: Define an objective function or metric to evaluate the performance of candidate architectures. In the context of YOLO, this might include metrics like Mean Average Precision (mAP), latency and model size.

(3) Architecture Search: Employ a search algorithm to explore the defined search space and find the best-performing architectures. This search can be performed through techniques like reinforcement learning, evolutionary algorithms or gradient-based optimization.

(4) Evaluation: For each candidate architecture, evaluate its performance on a validation dataset to calculate the defined objective function.

(5) Pruning: Prune fewer promising architectures to focus on more promising ones, optimizing the search process.

(6) Fine-Tuning: After selecting the best architecture, fine-tuning to further the search process.

(7) YOLO Integration: Incorporate the optimized architecture into the YOLO framework for object detection.

YOLONAS aims to strike a balance between accuracy and efficiency, ensuring that the resulting architecture is suitable for real-time object detection task, as YOLO is known for its speed and efficiency. The architecture of YOLONAS is depicted in Figure 15 [64] [67].

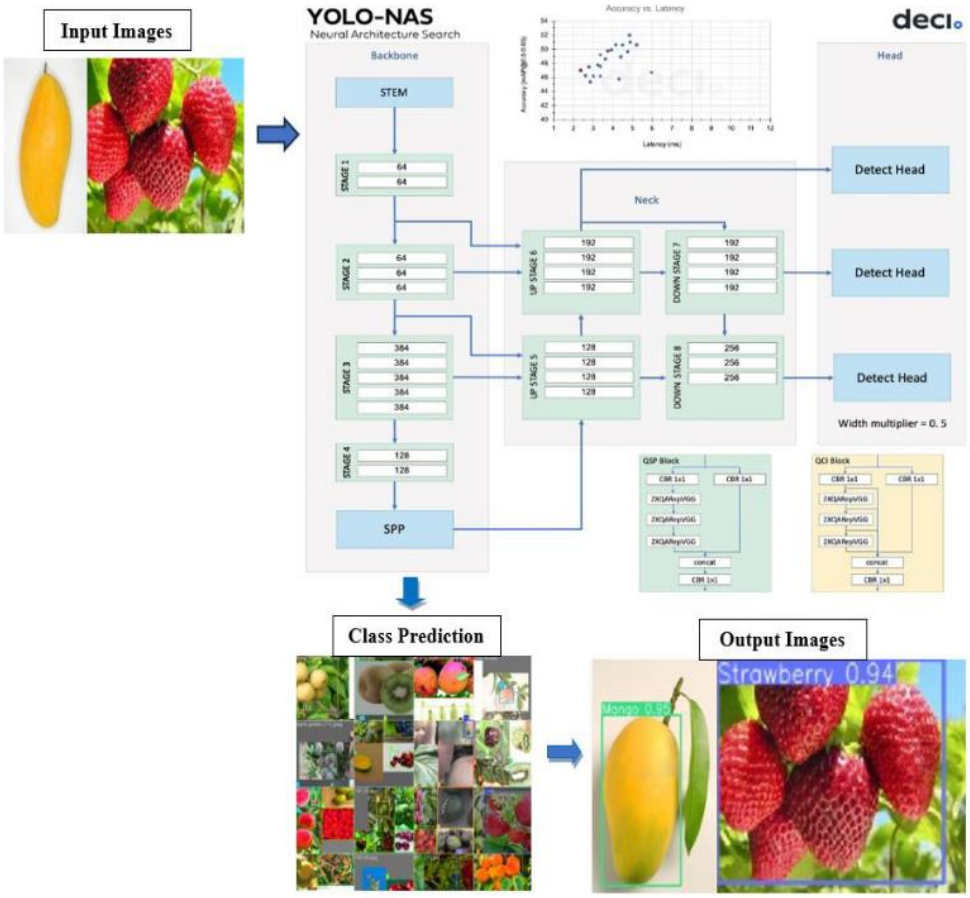

Figure 15: System Architecture of YOLONAS including the backbone, neck and head

YOLONAS consists of a backbone, neck, and head, which closely resemble several other models. The backbone algorithm derives distinctive characteristics from the input image. This model utilizes a dual-path backbone that consists of both dense feature and sparse feature fusion paths. The dense path consists of several convolutional layers that incorporate skip connections within dense blocks.

On the other hand, the sparse path includes transition blocks that decrease the spatial resolution and increase the number of channels. Both routes are interconnected with cross-stage partial connections, allowing for the movement of gradients. The neck layer amplifies the characteristics retrieved by the backbone, generating predictions at different scales. The YOLONAS system employs a multi-scale feature pyramid and backbone features to generate predictions at many scales, including small, medium and large.

The system's neck employs up-sampling and concatenation to merge data from different levels, hence expanding the receptive field. The head is responsible for carrying out the last stages of classification and regression tasks. The YOLONAS head is composed of two branches that run parallel to each other. One branch is dedicated to classification, while the other branch focuses on regression and utilizes generalized IoU loss. The classification branch determines the likelihood of each anchor box belonging to a specific class, while the regression branch calculates the precise coordinates of the bounding box.

In the realm of object detection, YOLO and YOLONAS represent significant advancements, each with distinct architectural and performance characteristics. YOLO is renowned for its straightforward, single-stage detection approach, featuring a backbone, neck and head designed for rapid object detection. Its architecture involves a fixed set of layers with manually tuned hyperparameter, striking a balance between speed and accuracy suitable for real-time applications. However, its standard quantization methods can impact model accuracy and efficiency. In contrast, YOLONAS incorporates a dual-path backbone with dense and sparse feature fusion paths, enhancing its ability to detect small objects and improve localization accuracy. It leverages advanced techniques such as NAS to automate and optimize network design, allowing for customized and efficient model architectures. Also, YOLONAS utilizes hybrid quantization methods and advanced training schemes, including quantization-aware modules (QSP and QCI) to minimize accuracy loss during post-training quantization.

While YOLO prioritizes speed, YOLONAS aims to optimize the balance between accuracy and efficiency, making it particularly well-suited for real-time-edge-device applications. Its integration of sophisticated quantization and optimization techniques, such as AutoNac for automated design, further distinguishes it from YOLO by improving both performance and adaptability. Overall, YOLONAS represents a significant evolution in object detection technology, combining advanced architectural features with cutting-edge optimization strategies to surpass the capabilities of traditional YOLO models.

3.3.4. Stratified Data Splitting

Stratified sampling is a data-splitting strategy that ensures that the proportion of each class in the training and test sets is the same as that in the original dataset. This approach is particularly useful when dealing with imbalanced datasets, where one class is significantly more prevalent than the others. Later, in section Additional Analysis, in subsequent section Training YOLOv5 Model with Stratified Data Splitting, we will apply this technique to further evaluate results and demonstrate its impact on the performance of the model.

CADEX and its extension DUPLEX are methods that select samples based on their mutual Euclidean distance. They start with two most distanced samples from the dataset T and then repetitively select samples with maximal distance to the previously sampled examples. Algorithm 3 which is presented in Figure 16 describes DUPLEX in more detail. These methods ensure a maximum coverage of T which represents dataset [68].

Algorithm 3 DUPLEX

1. Input: dataset T
2. For each $T_i \in \{T_{tr}, T_v, T_t\}$:
   2.1. Find $x_j, x_k \in T$ with the maximum mutual distance $\| x_j - x_k \|$.
   2.2. Extract x_j, x_k from T and add them to T_i.
3. Repeat:
   For each $T_i \in \{T_{tr}, T_v, T_t\}$:
   3.1. Find $x \in T$ with the maximum distance $\| x - s \|$ to the nearest previously sampled $s \in T_i$.
   3.2. Extract x from T and add it to T_i.

Figure 16: DUPLEX Algorithm

The basic idea of the stratified sampling is to explore the internal structure and distribution of the dataset T and exploit it to divide T into relatively homogeneous groups of samples (strata, clusters). The samples are then selected separately from each cluster. Stratified sampling is advantageous for the datasets, which can be divided into separate clusters. In such case, all regions of the input space may be adequately covered by the training subset Ttr and the estimate of the model performance measured on Tt will be highly precise.

On the other hand, for nearly uniformly distributed datasets, the stratified sampling will be comparable to Simple random sampling (SRS). Various clustering algorithms can be used to divide T into clusters including c-means or self-organizing maps. However, most of the clustering algorithms are sensitive to the choice of initial parameters, such as the desirable number of clusters or the choice of input variables. An inadequate setup of these parameters affects heavily the quality of the sampling and another important question is how to select samples from each cluster. There are two most common principles- a) select one sample from each cluster or b) randomly divide each cluster into the subsets.

The algorithm used for Stratified Data Splitting in scikit-learn does not necessarily select just one sample from each cluster. Instead, it aims to maintain the proportion of samples from each class within the subsets. Firstly, it calculates the class distribution in the entire dataset, then when it

split the data into subsets (like train and test sets, or in our case 25%, 50%, 70% of the data), it ensures that each subset has a similar distribution of classes as the original dataset. While it maintains these proportions, the exact samples allocated to each subset can be random within the constraints of keeping the class proportions intact. So, in essence, it's not strictly "one sample from each cluster", but rather a process that maintains the relative representation of classes in each subset while allowing for randomness in the selection of samples within those constraints.

3.3.5. Size Models of YOLOv5

The YOLOv5 provides different models with different configurations and parameter sizes (Figure 17). Respectively the YOLOv5s, YOLOv5m, YOLOv5l and YOLOv5x mean small (s), medium (m), large (l) and extra-large (x) models [67]. Each model has its pros and cons, but eventually, the differences are in their complexities, performances and overall accuracy. From [67] results show that the size of the models varies from 14MB to 168MB, mAP values for the full COCO dataset is from 36. To 50.1 and the inference time on the Nvidia V100 GPU varies from 2.2 to 6.0 milliseconds.

.

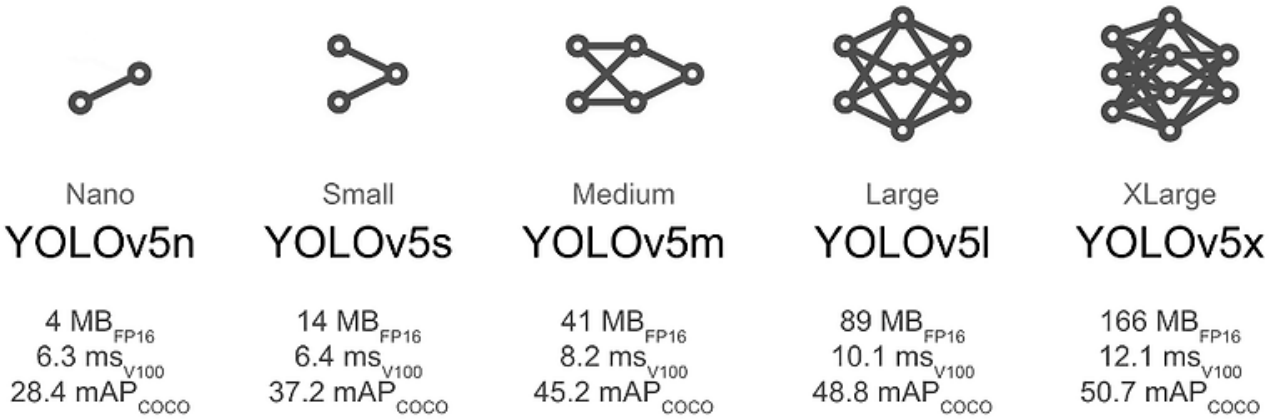

Figure 17: YOLOv5 different model sizes, where FP16 stands for the half floating-point precision, V100 is an inference time in milliseconds on the Nvidia V100 GPU and mAP based on the original COCO dataset.

The smallest version, YOLOv5s is designed to be lightweight and fast. It has fewer parameter and smaller model size, making it suitable for applications where computational resources are limited, such as mobile devices or real-time detection on embedded systems. However, due to its simplicity, it may not achieve the same level of accuracy as its larger counterparts.

YOLOv5m is a mid-sized version that strikes a balance between speed and accuracy. It has more parameters than YOLOv5s, offering improved detection capabilities, especially for more complex or varied datasets. This version is ideal for scenarios where you need a good trade-off between computational efficiency and performance.

The largest version, YOLOv5l has the most parameters and the highest model complexity. This version provides the best accuracy among the three, making it suitable for applications where detection quality is paramount, such as detailed image analysis or tasks requiring high precision. However, this comes at the cost of increased computational demands, which means it requires more powerful hardware and longer training times. Furthermore, the choice between small, medium and large size of YOLOv5 depends on the specific requirements of given application.

As we already mentioned, the backbone (CSPNet-based) is responsible for feature extraction. In YOLOv5s, fewer layers and channels are used in the backbone compared to YOLOv5m and YOLOv5l, which have deeper architectures and more detailed features. The neck (which includes the PANet and FPN for feature aggregation) also differs in depth and width across the models. YOLOv5l has the most complex neck with deeper feature pyramid levels, while YOLOv5s uses a simplified version.

In summary, the main differences in architecture between the small, medium and large models of YOLOv5 lie in the number of layers, parameters, feature extraction and aggregation capacity, affecting both performance and computational cost. As the model size increases, so does the ability to learn finer details in object detection, especially for smaller objects, complex scenes and high-resolution images.Version small is best for speed, medium offers a balance, while large excels in accuracy. Understanding these trade-offs is essential when selecting the appropriate model size for specific object detection task.

3.3.6. Performance Metrics

In object detection tasks, evaluating the performance of models like YOLOv5 requires a deep understanding of metric such as mean Average Precision (mAP), Precision and Recall. These metrics are crucial in determining how well a model detects objects and distinguishes between different classes. Precision is the metric that quantifies how many of the objects detected by the model are actually relevant. In other words, it measures the accuracy of the objects detected by the model are actually relevant. In other words, it measures the accuracy of the positive predictions. High precision means that the model produces fewer false positives-instances where the model identifies an object that is not present.

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

Recall, on the other hand, measures how well the model identifies all relevant objects. It is the ratio of correctly identified objects (true positives) to the total number of actual objects in the dataset (true positives and false negatives). High Recall indicates that the model can detect most of the objects, but it may also include more false positives.

$$True\ positives = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

Furthermore, mAP is a comprehensive metric that combines Precision and Recall across different thresholds. It is calculated as the mean of the average precision values for each class. In object detection, mAP is computed over a range of Intersection over Union (IoU) thresholds, which measure the overlap between the predicted bounding box and the ground truth bounding box. Also, mAP provides a single value that summarizes the model's ability to balance Precision and Recall. A higher mAP indicates better overall performance, with the model effectively identifying objects with both high accuracy and coverage.

Precision emphasizes the accuracy of the model's predictions, while Recall focuses on its ability to capture all relevant objects. There is often a trade-off between these two metrics like improving Precision can reduce Recall and vice versa.

While Precision and Recall provide insights into specific aspects of the model's detection capabilities, mAP offers a comprehensive evaluation, making it an essential metric for comparing different object detection models. These metrics will guide the analysis of the chosen object detection models experiments and the subsequent comparison between them.

When working with object detection models like YOLO, YOLONAS and Detectron2, one of the key metrics used to evaluate and compare their performance is the mAP (mean Average Precision). Since both YOLO and YOLONAS natively provide mAP as a primary performance metric, it is essential to calculate and utilize mAP for Detectron2 to enable a consistent and fair comparison between these three models. Also, mAP offers a balanced view of both Precision and

Recall across different IoU thresholds and object sizes, making it a more comprehensive metric for comparing model effectiveness. In Detectron2, mAP is calculated as the average of the AP values across different IoU. Metric mAP is computed by taking the mean of AP values across all classes, areas and IoU thresholds like in an equation:

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i$$

Where N is the number of IoU thresholds and AP_i is the Average Precison at the ith IoU threshold.

Iterations and epochs refer to different aspects of the training process in machine learning, but they are related and understanding their relationship can help with comparisons. Since YOLO and YOLONAS provide number of epochs and Detectron2 provides number of iterations, we will converse iterations into epochs for easier comparison. One epoch refers to one complete pass through the entire training dataset. This means, when the model has seen every training sample once, it completes one epoch. On the other hand, an iteration refers to one update of the model's parameters. An iteration typically happens after processing one batch of data. The relationship between epochs and iterations depends on the batch size and the size of the dataset:

$$Iterations\ per\ Epoch = \frac{Total\ Number\ of\ Samples}{Batch\ Size}$$

So, in model Detectron2 the conversion between iterations and epochs have been calculated in order to get better understanding of results between these models.

4. Results

In this section, we will present the results obtained from training and evaluating different object detection models which I mentioned before. All models were implemented and trained using Python, with the experiments conducted in both Jupyter Notebook software and Google Colaboratory environments. The performance of each model was assessed on a dataset processed through various techniques to enhance detection accuracy.

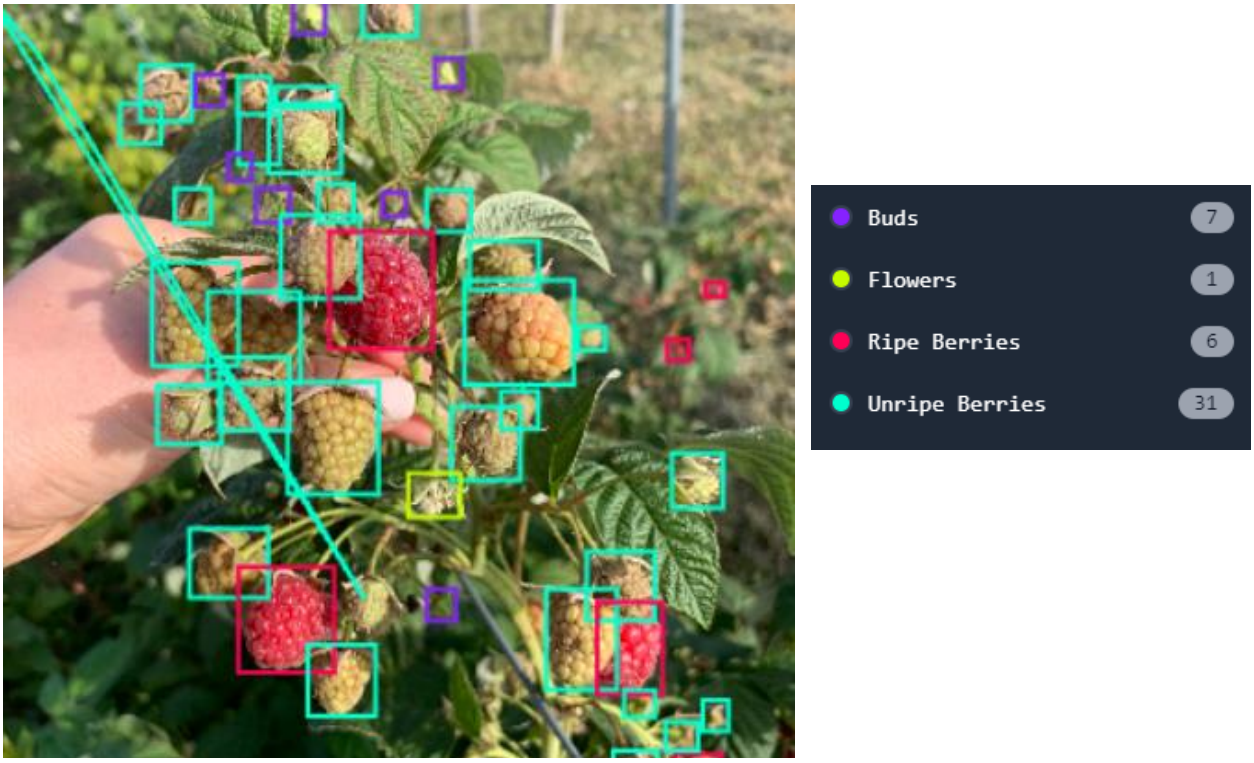

Figure 18: Example of detected classes of raspberries

Figure 18 illustrates a representative image displaying the detected classes of raspberries after training the models on the dataset. The detected classes-buds, flowers, ripe and unripe berries are presented in the image, highlighting the model's ability to distinguish between different growth stages of the raspberry plants. Each class exhibits unique characteristics in terms of texture, shape, color and size which the models were able to detect. In addition to the classes, the image also contains visible elements of the background and greenery, as expected in real-world agricultural environments. Despite these complexities, all the models in this study were able to detect the different classes with similar level of accuracy as depicted in this image. Each class is enclosed in a bounding box of a distinct color, providing a clear visual representation of the detections. This color-coded approach allows for easy differentiation between the classes across all models, ensuring consistency in the detection process.

First model is YOLOv5, where we specifically leverage the small architecture to suit the characteristics of my dataset. Later, in section Comparison between different sizes of YOLOv5 model I analyzed different sizes of architecture of this model, from small, medium to large. To optimize the model's performance, we conducted a series of experiments by adjusting key hyperparameters, including IoU threshold, learning rate, batch size, number of epochs.

The impact of these adjustments was evaluated using performance metrics such as mAP, Precision and Recall. Results of different batch sizes, number of epochs and their performances is shown in Table 1 below.

Batch Size	Number of Epochs	Training Time	mAP per classes	mAP overall	Precision	Recall
4	10	20 min	Buds = 58%, Unripe B. = 82% Flowers =58%, Ripe B. = 11%	52%	79%	43%
4	30	41 min	Buds = 73%, Unripe B. = 87% Flowers =67%, Ripe B. = 46%	68%	79%	59%
8	60	1h 10min	Buds = 71%, Unripe B. = 86% Flowers =64%, Ripe B. = 76%	75%	78%	69%
16	40	37 min	Buds = 72%, Unripe B. = 87% Flowers =64%, Ripe B. = 77%	74%	77%	66%
16	60	48 min	Buds = 72%, Unripe B. = 87% Flowers = 64%, Ripe B. = 77%	75%	78%	78%
16	80	1h 7 min	Buds = 73%, Unripe B. = 88% Flowers =66%, Ripe B. = 76%	76%	79%	68%
16	100	1h 26 min	Buds = 68%, Unripe B. = 85% Flowers =63%, Ripe B. = 77%	73%	78%	66%

Table 1: YOLOv5 Results for different parameters

Initial setup of batch size 4 and 10 epochs provided a baseline for the model's performance with minimal training. While precision was high at 79%, the recall of 43% indicated the model missed a significant number of true positives. Doubling the batch size and extending the training further enhanced mAP and recall, though there was a slight decrease in precision, indicating a trade-off between accuracy and detection capacity. With and increase batch size of 16 and remaining number of epochs of 60 yielded balanced improvements in both precision and recall, with both metrics reaching 78%. The consistent mAP indicated this setup as optimal in terms of time and performance balance.

At 100 epochs, the model showed a decrease in mAP and recall, suggesting potential overfitting or that the model reached a plateau where additional training did not further enhance performance. Based on these experiments, the most effective configuration for Raspberry dataset was with a batch size of 16 and 60 epochs, offering an optimal balance of mAP, precision and recall, while also being efficient in terms of training time. Furthermore, this configuration will be compared with other models.

Next model is Detectron2, where the focus was on detecting small objects, especially ripe and unripe berries, which showed the lowest Precision. Despite the robustness of Detectron2, several challenges were encountered, particularly with the model's performance on small objects and specific classes within the dataset.

Number of iterations	Number of epochs	Training Time	Precision	Precision for different areas	Recall	Recall for different areas
500	16	24 min	27%	Small = 9% Medium = 16% Large = 70%	19%	Small = 8% Medium = 14% Large = 70%
700	22	33min	39%	Small = 18% Medium = 24% Large = 50%	32%	Small = 20% Medium = 26% Large = 50%
900	29	43min	54%	Small = 32% Medium = 37% Large = 70%	48%	Small = 32% Medium = 37% Large = 70%
1500	48	1h 14min	67%	Small = 45% Medium = 53% Large = 60%	63%	Small = 50% Medium = 59% Large = 60%
2000	63	1h 39min	55%	Small = 36% Medium = 40% Large = 70%	49%	Small = 34% Medium = 39% Large = 70%
3000	95	2h 5min	60%	Small = 41% Medium = 48% Large = 70%	55%	Small = 39% Medium = 45% Large = 70%

Table 2: Detectron2 Results for different parameters

In Detectron2, AP refers to Average Precision, which is not Precision or mAP. It is a metric used to evaluate the performance of object detection models on a per-class basis. AP is calculated by finding the area under the Precision-Recall curve for a particular class. Due to the other two models using Precision and Recall, we will convert AP and AR into these metrics for further comparison. Precision for areas like small, medium and large refers to evaluating objects based on their area in the image and average across IoU thresholds from 0.50 to 0.95. So, if model needs to detect small objects, Precision will give insight into its performance in that context.

Small objects are typically more challenging to detect, so a low Precision here in Table 2 indicate that the model struggles with finer details, which indicates that this Raspberry dataset is not showing good results with this model. Recall in Detectron2 is another important metric used to evaluate the performance of object detection models, implementing metrics like Precision. It measures the ability of a model to detect all relevant objects in an image. Recall gives a sense of how well the model is capturing all true positives, which means how well it is finding all the objects that it should detect.

As we mention before in section Performance Metrics, model Detectron2 provide number of iterations, while other two models provide number of epochs.

For better comparing, we did a conversation from number of iterations into number of epochs for total number of images and a batch size of 64. So, after conversation formula we got approximately 31,84 iterations per epoch.

As we mentioned, the dataset primarily consisted of small objects, which presented a significant challenge during training. Precision for small objects was notably low throughout the training and evaluating process. As I previously noted, the small size of the berries could exacerbate these issues, particularly for a class where fine distinctions are crucial.

Around the 1500 iteration or 48th epoch, metrics did not again show good results, but it was the best performance comparing to other number of epochs, and after increasing number of epochs there was a slight decrease in all metrics, where overfitting probably occurred. After 29th epoch, some improvement was observed, but the Precision for ripe berries remained low compared to other classes. This suggests that more targeted strategies, such as increasing the representation of ripe berries in the training set or employing advanced techniques like focal loss, might be necessary to address this issue.

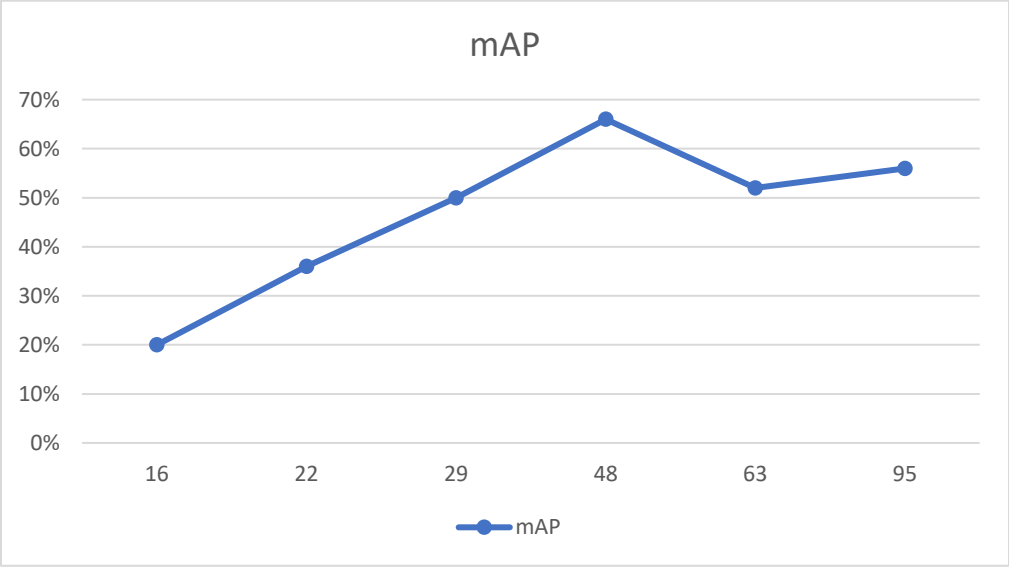

Figure 19: mAP metrics for different number of iterations in Detectron2

Model Detectron2 does not naturally gives results of mAP, so we needed to calculate it from formula in section Performance Matrics. So, Figure 19 represents mAP for different number of iterations in Detectron2 Model. For epochs 16, 22, 29, 48, 63, 95 mAP value were approximately 20%, 36%, 50%, 66%, 52%, 56% respectively. Moreover, again about 48th epoch showed the best performance for this data.

Also, mAP is being worse than some specific values of AP and usually indicates that the model's performance is not consistent across different IoU thresholds or object sizes. The mAP metric is more holistic, so it tends to be lower if there's a wide variance in performance across the IoU thresholds or object sizes. Also, for improving mAP metric, it is good to focus on improving the model's performance at higher IoU thresholds or across all object sizes, rather than optimizing for one specific case.

Third model is YOLONAS, demonstrated strong performance from the early stages of training. The model was trained using different epoch settings to evaluate its performance across varying training durations. The objective was to balance Precision, Recall and mAP while ensuring the model's generalization ability on unseen data. The results are shown in Table 3.

Number of epochs	Training Time	mAP	Precision	Recall
25	1h 18min	84%	77%	96%
50	2h 30min	85%	82%	96%
60	3h 4min	87%	83%	97%
75	3h 50min	85%	82%	97%

Table 3: Performance on different number of epochs for YOLONAS Model

After 25 epochs, the model exhibited a high recall and commendable mAP, reflecting its strong ability to accurately detect a significant proportion of actual objects. However, precision was somewhat lower, indicating potential for improvement in minimizing false positives. Extending the training to 75 epochs led to an initial improvement in all key metrics, with the best performance observed around the 60^{th} epoch. Precision increased significantly, indicating better accuracy in identifying true positives. So, around 60^{th} epoch, all metrics are showing the best performance, where mAP is 87%, Precision 83% and Recall 97%.

These findings suggest that for this dataset, the optimal number of epochs for training YOLONAS lies between 50 and 60, balancing training time and model performance.

4.1 Additional Analysis

In this Section, we aim to delve deeper into the YOLOv5 model, exploring how various changes and experiments impact its performance on the Raspberry dataset. The goal is to gain a better understanding of how the model behaves under different conditions and to optimize its performance for this specific application. Two primary analyses are conducted within this additional exploration: Comparison between different sizes of YOLOv5 models and training YOLOv5 models with Stratified data splitting.

First section investigates how the performance of the YOLOv5 model varies with different model sizes-small, medium and large. By comparing these sizes, we aim to determine the most suitable model size for accurately detecting raspberries while balancing training efficiency. In second analysis, I experiment with stratified data splitting to see how it affects the training process and model performance. Stratified splitting ensures that each subset of the data maintains the same distribution of classes, which may lead to more robust model training and better generalization.

These analyses will provide insights into the effectiveness of various YOLOv5 configurations, offering guidance on the best practices for training and deploying this model on the raspberry dataset.

4.1.1. Comparison between different sizes of YOLOv5 Model

The YOLOv5 provides different models with different configurations and parameter sizes. Respectively the YOLOv5s, YOLOv5m, YOLOv5l and YOLOv5x mean small (s), medium (m), large (l) and extra-large (x) models. Each model has its pros and cons, but eventually, the differences are in their complexities, performances and overall accuracy. All the differences, model architecture and additional details regarding YOLOv5 model sizes are thoroughly explained in Section 3.3.5., Size Models of YOLOv5.

In this section, I will explore and compare the performance of small (s), medium (m) and large (l) model sizes of YOLOv5 on the Raspberry dataset. Each model was trained under various configurations, including batch size and the number of epochs, to evaluate how these factors impact the model's mAP and training time.

These details are crucial for understanding the trade-offs between model complexity, accuracy and computational time.

Table 4 represents Results between small, medium and large sizes of YOLOv5 Model on a batch size 16 and different number of epochs.

Batch size	Number of epochs	Size of Model	mAP	Time
16	40	s	74%	37min
16	40	m	76%	43min
16	40	l	78%	1h 5min
16	60	s	75%	48min
16	60	m	75%	1h 7min
16	60	l	80%	1h 38min
16	80	s	74%	1h 7min
16	80	m	77%	1h 26min
16	80	l	81%	2h 9min
16	100	s	73%	1h 26min
16	100	m	76%	1h 38min
16	100	l	79%	2h 44min

Table 4: Results between small, medium and large sizes of YOLOv5 Model

With 40 epochs, the large model outperforms the medium and small models, achieving the highest mAP of 78%. The training time increases with model size, as expected, with the large model taking nearly double the time of the small model.

However, the difference between small and large models is relatively modest (4%), suggesting that the large model provides slightly better accuracy but at the cost of significantly longer training time. When the number of epochs is increased to 60, the large models further improve its performance to a mAP of 80%, while the small and medium models show little change in mAP (both at 75%). The large model's training time also increases to 1 hour and 38 minutes. This indicates that the large model benefits more from additional epochs, improving accuracy while the smaller models hit a plateau.

With 80 epochs, the large models reach a mAP of 81%, while the medium model slightly improves to 77%, and the small model remains at 74%. The training times continue to increase, especially for the large model, which takes more than 2 hours. The small model's performance begins to decline slightly, suggesting that it may be overfitting as the number of epochs increases. At 100 epochs, the large model's mAP decreases slightly to 79%, while the medium and small models also show slight declines in mAP.

The training times reach their peak, with the large model taking nearly 3 hours. The decrease in mAP for all models suggests diminishing returns with further increases in the number of epochs, possibly due to overfitting.

Based on the analysis of different YOLOv5 model sizes trained on the Raspberry dataset, it is evident that the large YOLOv5 model consistently achieves the highest accuracy, with a mAP reaching up to 81% at 80 epochs.

This model benefits the most from an increased number of epochs, showing a clear improvement in detection performance, especially compared to the small and medium models.

However, this higher accuracy comes with significantly longer training times, reaching up to nearly 3 hours at 100 epochs. The small model, while offering the fastest training times, tends to overfit as the number of epochs increases and shows limited accuracy improvement, making it less suitable for tasks requiring high precision. The medium model provides a middle ground, with decent accuracy and moderate training times, but it still does not match the performance of the large model.

The best model to train and analyze the Raspberry dataset is the large YOLOv5 model, especially if accuracy is the top priority. It consistently outperforms the smaller models in detection performance, making it the most reliable choice for analyzing data. However, if time efficiency is a concern, the medium model can be a reasonable alternative, offering a balance between training time and accuracy.

4.1.2. Training YOLOv5 Model with Stratified Data Splitting

In this section, we have delved into the process of training a YOLOv5 neural network for object detection using different percentages of the dataset, employing the Stratified Data Splitting technique from scikit-learn library for Python.

As we already proceed our dataset which is unbalanced, the goal is to see if this technique which will decrease number of images and their annotations will show us some information and further discussion about performances and future work. This section provides a detailed overview of the training process and performance of the YOLOv5 model are different data percentages (25% of the data, 50% of the data, 75% of the data and 100% of the data), highlighting the effectiveness of the Stratified Data Splitting technique.

We have used parameters that showed the best performance and accuracy on a previously trained 100% dataset model, which is small architecture of a YOLOv5 model (YOLOv5s) and the batch size is 16 and the number of epochs is 60.

The following results obtained at each stage, including total number of images and annotations, percentage of annotations per class, time and metrics such as mean Average Precision (mAP) are shown in Table 5.

	Total number of images	Total number of annotations	Percentage of annotations per class	Training time	mAP
25%	510	11 650	Buds= 2745 (23.56%) Flowers= 1115 (9.57%) UB= 7632 (65.51%) RB= 158 (1.36%)	12min	**60.4%** B=62.3% F=59.3% UB=82.2% RB=37.6%
50%	1019	23 012	Buds= 6071 (26.38%) Flowers= 2363 (10.27%) UB= 14352 (62.36%) RB= 227 (0.99%)	25min	**67.7%** B=70.9% F=61.4% UB=84.6% RB=32.6%
75%	1528	34 492	Buds= 9041 (26.21%) Flowers= 3630 (10.52%) UB= 21517 (62.38%) RB= 304 (0.88%)	34min	**71.2%** B=70.9% F=60.5% UB=84.9% RB=68.5%
100%	2038	46 142	Buds= 11786 (25.54%) Flowers= 4745 (10.28%) UB= 29149 (63.17%) RB= 462 (1.01%)	48min	**75%** B=72% F=64% UB=87% RB=77%

Table 5: Results of Stratified Data Splitting

As the percentage of data used for training increases, there is a noticeable improvement in the model's mean Average Precision (mAP): from 60.4% with 25% of the data to 75% with 71.2% and finally to 100% with 75%, we can see a consistent upward trend. This indicates that with more data, the model becomes more effective at accurately detecting objects in the images.

Across all percentages of data, the model shows varying levels of precision for different classes. Unripe berries (UB) consistently has the highest precision, reaching 87% with 100% of the data, also buds and flowers show improvements in precision as more data is used for training. However, ripe berries (RB) remains the most challenging class for the model, with lower precision values across the board.

The increase in the number of images and annotations from 25% to 100% of the data is evident and this increase leads to more diverse examples for the model to learn from, which is reflected in the improved performance metrics. It is noteworthy that the training time increases as more data is used which is expected, as training with a larger dataset requires more computation.

As a conclusion for this experiment, the results show that the YOLOv5 model benefits significantly from a larger amount of data. The class unripe berries remains the class with the highest precision, while ripe berries remains the most challenging. Furthermore, balancing between training time and model performance, using a larger percentage of the dataset seems to be beneficial for raspberry object detection task.

Also, as we have seen in previously object detection models, class ripe berries is the most challenging class for the models. Although its specific size, shape and color is different from other classes, the main factor for low accuracy is that this class has the smallest percentage of annotations comparing other classes and it often has confusion with background.

5. Discussion

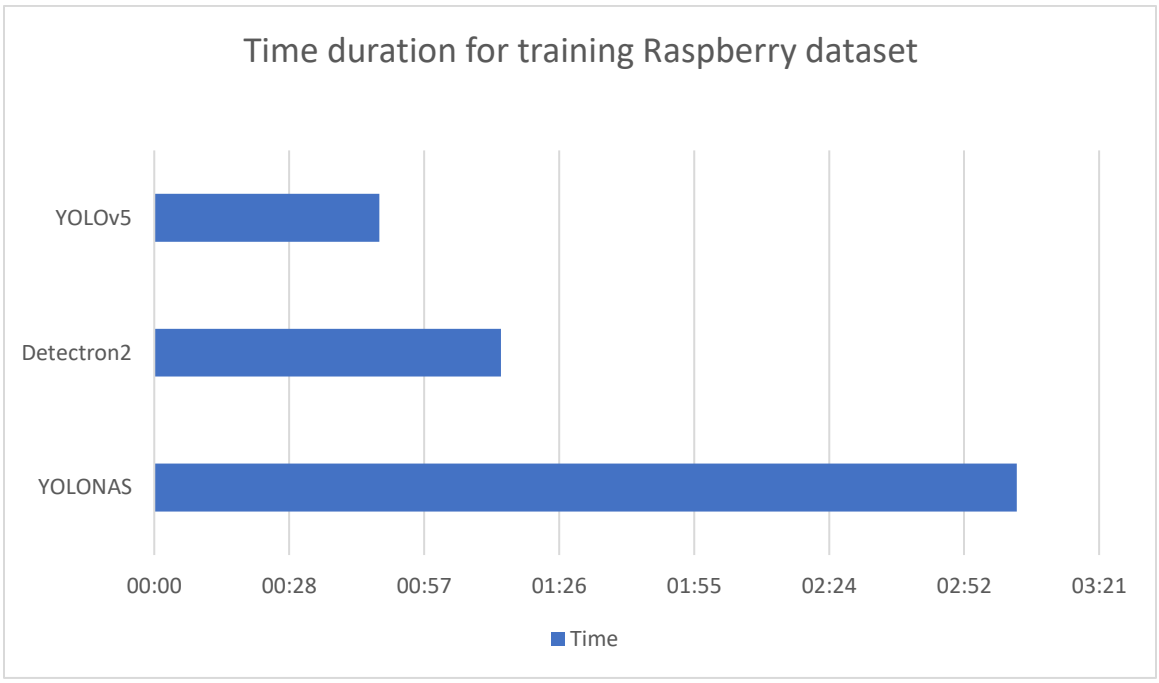

Figure 20: Time duration for training and validation of data between models

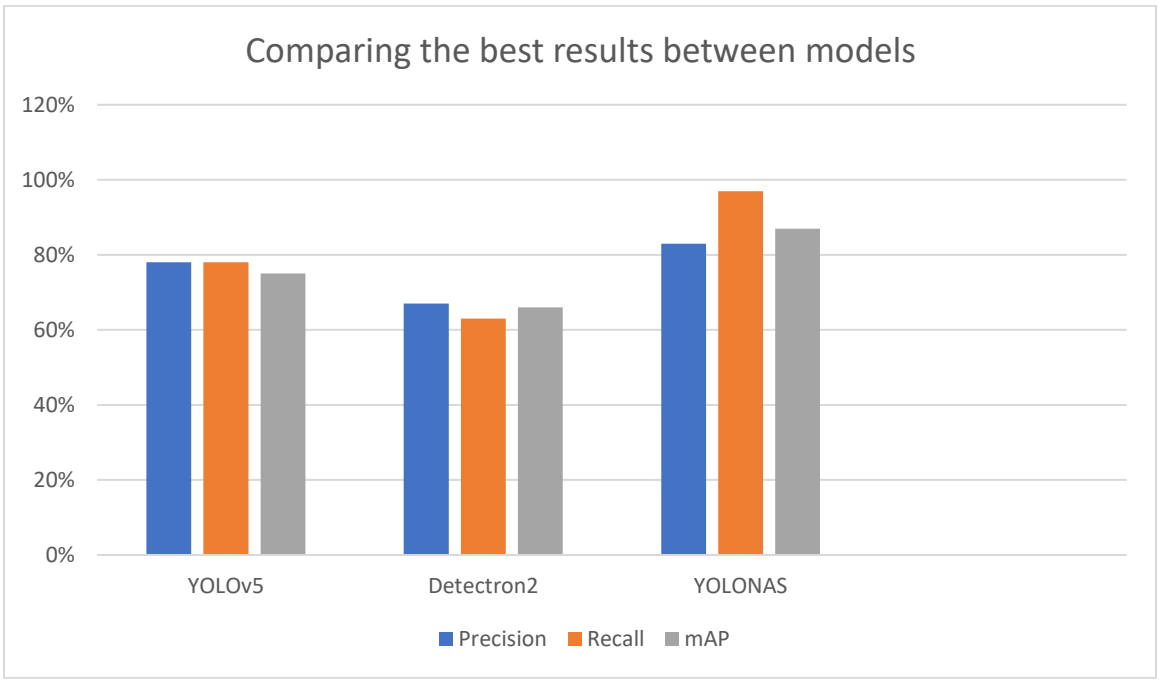

Figure 21: Comparison data performance between models

Figure 20 represent the time taken for training and this raspberry dataset on different models. Figure 21 represents three chosen object detection models for this raspberry data and their performance, based on mAP, Precision and Recall metrics. These metrics provide insight into each model's ability to accurately detect objects, particularly focusing on the detection of raspberries. YOLOv5 exhibits a well-balanced performance with a mAP of 75%, signifying that the model achieves strong average detecting accuracy across various classes.

The Precision and Recall values are also well-matched at 78%, suggesting that YOLOv5 maintains a good balance between correctly identifying objects and minimizing false positives and false negatives. Also, the model's relatively short training time of 48 minutes makes it an efficient option, particularly in scenarios where quick model deployment is essential.

Detectron2 shows the weakest performance among the three models, with a mAP of 66%, Precision of 67% and Recall of 63%, The lower Recall value struggles to detect all relevant objects, leading to more false negatives. This underperformance could be particularly problematic for detecting smaller objects like raspberries. Detectron2 takes 1h and 14 minutes to rain, which is longer than YOLOv5, but sill less than YOLONAS. The combination of lower accuracy and longer training time makes Detectron2 a less attractive option for this dataset. YOLONAS is the top performer, with mAP of 87%, Precision of 83% and an exceptionally high Recall indicates that YOLONAS is extremely effective at detecting nearly all relevant objects, with very few false negatives. The model's slightly lower Precision compared to its Recall suggests a minor trade-off with slightly more false positives, but this is offset by its overall high detection accuracy.

However, this performance comes at the cost of significantly longer training time-3 hours and 4 minutes. This extended training period might be a consideration in time-sensitive applications, but the accuracy gains are substantial, making YOLONAS the most suitable model for detecting raspberries in this dataset.

Also, in model Detectron2 the ripe berries class was particularly problematic during the training process. Up until approximately 900^{th} iteration, the model failed to recognize this class entirely. This might be the problem of class imbalance, because ripe berries have significantly smaller number of instances then unripe berries, so the model might have struggled to learn the features that distinguish ripe berries. Also, the ripe berries may share similar features with unripe ones, causing the model to confuse the two classes.

This model provided a strong starting point for object detection in the raspberry dataset, the model struggled with the detection of small objects and the accurate classification of ripe berries. These challenges highlight the need for further refinement in the model architecture and training strategy, particularly when dealing with datasets that include small, visually similar objects. Future work will focus on addressing these issues through improved data augmentation, architecture adjustments and potentially integrating specialized small object detection techniques.

In YOLONAS model after the 60^{th} epoch, a slight decline in metrics was observed, which is likely due to overfitting, where the model starts to perform well on the training data at the

expense of its ability to generalize to new data. The results indicate that YOLONAS can effectively handle the raspberry data set, achieving high Recall and mAP, particularly when trained for a moderate number of epochs. Training for 25 epochs provides a good baseline with reasonable performance, while extending the training to around 60 epochs further refines the model's Precision and mAP. However, care must be taken to monitor the model's performance beyond this point to prevent overfitting, as observed with the slight decline after the 60^{th} epoch.

Throughout the experiments, the class unripe berries consistently exhibited the highest precision across all models. This suggest that this class are relatively easy for the models to detect, likely due to their distinct visual characteristics compared to other classes. This high precision is beneficial for applications focused on early-stage crop monitoring, where the detection of unripe berries is crucial for predicting harvest times and managing resources.

In contrast, the class ripe berries proved to be the most challenging for all models, with lower precision values observed across the board. This could be due to the similarity in appearance between ripe berries and other objects or backgrounds in the images, leading to higher confusion rates. Additionally, the significantly smaller number of labeled instances for Ripe Berries-463 instances compared to 29,156 for unripe berries, 11,788 for buds and 4,747 for flowers could have contributed to the model's difficulty in accurately detecting ripe berries. The imbalance in the dataset may have resulted in the models having less information to learn from for this class, thus impacting their performance. Improving the detection of ripe berries is a key area for future research, as accurate identification of this class is critical for optimizing harvest timing and ensuring fruit quality.

6. Conclusion

This master thesis has explored the application of state-of-the-art deep learning models: YOLOv5, Detectron2 and YOLONAS for the detection and classification of raspberries. The primary objective was to develop and evaluate robust models capable of accurately identifying different stages of raspberry growth, including buds, flowers, unripe berries and ripe berries. Through systematic experimentation and analysis, we aimed to determine the strengths and limitations of each model, as well as the influence of various training strategies on model performance. This research contributes to the broader field of precision agriculture by enhancing my understanding of how deep learning techniques can be applied to crop monitoring.

The three models-YOLOv5, Detecton2 and YOLONAS demonstrated unique strengths in the context of raspberry detection. The YOLOv5 model, particularly small architecture (YOLOv5s) achieved a mean Average Precision of 75% in 48 minutes, which is indicative of its capability to quickly process images while maintaining a reasonable level of accuracy. This makes it particularly suitable for applications where real-time or near-real-time analysis is required. Detectron2, another model tested in this research, achieved a mAP of 66% in 1 hour and 14 minutes. While it did not reach the same level of accuracy as YOLOv5s, it offered valuable insights, particularly in terms of handling complex image features and backgrounds. Detectron2's slightly longer processing time and lower mAP suggest that it may be more suitable for scenarios where accuracy can be slightly compromised in favor of better handling of diverse environmental conditions. YOLONAS model emerged as the most accurate of the three, with mAP of 87% in 3 hours and 4 minutes. This higher accuracy, however, comes at the cost of longer training times, and time is less of a constraint. The extended training time is justified by its superior performance, especially in scenarios requiring fine-grained classification of Raspberry growth stages.

In addition to comparing different models, this thesis also investigated the impact of model size within the YOLOv5 family (small, medium, large) and the effect of stratified data splitting on training outcomes. The size of the YOLOv5 model has a direct influence on its performance, with larger models generally providing better accuracy at the expense of longer training times. For instance, the YOLOv5 large model achieved a mAP of 81% in 2 hours and 9 minutes. This represents an improvement in accuracy compared to the small architecture, though it requires more than double the training time.

The results suggest that for applications where the highest accuracy is paramount and time is less of a concern, the larger YOLOv5 models are more appropriate. Conversely, for time-sensitive tasks, the small or medium models provide a good balance between speed and accuracy. Stratified data splitting, where the training data is divided to ensure that each subset contains a representative sample of all classes, was also explored.

The results showed that using 100% of the data resulted in a mAP of 75% in 48 minutes for the YOLOv5s model, while using 75% of the data led to a slightly lower mAP of 71% in 34 minutes. This indicates that while reducing the amount of training data can speed up training, it may

slightly reduce the model's accuracy. However, the impact is not drastic, suggesting that stratified data splitting can be an effective strategy when computational resources or time are limited.

The findings of this research underscore the potential of deep learning models in advancing precision agriculture. The dataset used for this research originated from Latvia, providing a specific regional context. However, the techniques and models developed could have broader applications. For example, future research could explore the application of these models to raspberry detection in Serbia, which is the third-largest producer of raspberries in the world. Adapting the models to the unique conditions and characteristics of Serbian raspberry farms could lead to significant advancements in yield prediction, harvest optimization and quality control.

Furthermore, there are several areas where future search could further enhance these models. One potential avenue is the exploration of additional data augmentation techniques, such as synthetic data generation or the inclusion of environmental variability (e.g., different lighting conditions, weather effects) in the training dataset. These techniques could improve the model's robustness and generalizability, making them more effective in real-world agricultural settings. Another area for future work is the integration of these detection models into real-time monitoring systems. By deploying these models on edge devices or integrating them with drone or satellite imagery, farmers could receive immediate feedback on crop conditions, allowing for more timely interventions. This real-time capability could be particularly valuable for large-scale farms or in regions where labor is scarce.

Moreover, the application of these detection techniques could be extended beyond raspberries to other crops with similar detection challenges. By adapting the models to different crops, this research could contribute to the development of more precise, efficient and scalable solutions for a wide range of agricultural applications, ultimately supporting sustainable agricultural practices and improving food security.

This thesis contributes to the growing body of knowledge in precision agriculture and highlights the potential for further advancements through the continued refinement of deep learning techniques. By building on the foundation laid in this thesis, future work can explore new methodologies and applications, ultimately leading to more efficient and effective crop monitoring systems that benefit both farmers and the broader agricultural industry.

Bibliography

[1] Maxine M. Thompson, *Survey of Chromosome Numbers in Rubus (Rosaceae: Rosoideae)*, Annals of the Missouri Botanical Garden, Published by Missouri Botanical Garden Press 1997, Vol.84, No.1 (1997), pp. 128-164, URL: https://www.jstor.org/stable/2399958?origin=crossref

[2] Gunars Lacis, Irita Kota-Dombrovska and Sarmite Strautina, *Evaluation of Red Raspberry Cultivars used for Breeding and Commercial growing in the Baltic region*, Institute of Horticulture, Latvia University of Agriculture, DOI:10.1515/prolas-2017-0034, January 2017

[3] Jason Brownlee, *Deep Learning for Computer Vision: Image Classification, Object Detection and Face Recognition in Python*, 2020, edition v1.8

[4] Jose Nranjo-Torres, Marco Mora, Claudio Fredes and Andres Valenzuela, *Disease and Defect Detection System for Raspberries Based on Convolutional Neural Networks*, Appl. Sci. 2021, 11(24), 11868; https://doi.org/10.3390/app112411868

[5] Tianyu Li, Yu Sun, *An Intelligent Food Inventory Monitoring System using Machine Learning and Computer Vision*, Conference: 3rd International Conference on Data Science and Machine Learning, September 2022, DOI: 10.5121/csit.2022.121512

[6] A. Nicholas E. Birch, Graham S. Begg, Geoffrey R. Squire, *How agro-ecological research helps to address food security issues under new IPM and pesticide reduction policies for global crop production systems*, Journal of Experimental Botany, Volume 62, Issue 10, June 2011, Pages 3251-3261, 01.06.2011.

[7] Feng Xiao, Haibin Wang, Yegin Xu and Ruiging Zhang, *Fruit Detection and Recognition Based on Deep Learning for Automatic Harvesting: An Overviev and Review*, Agronomy 2023, 13(6), 1625; https://doi.org/10.3390/agronomy13061625

[8] I. Augspole, F. Dimins, I.Romanova and A.Linina, *Characterization of red raspberry (Rubus idaeus L.) for their physicochemical and morphological properties*, Agronomy Research 19(S3), 1227-1233, May 13th 2021, https://doi.org/10.15159/AR.21.077

[9] Octavio Paredes-Lopez, Martha L. Cervantes-Ceja, Talia Hernandez-Perez and Monica Vigna-Perez, *Berries: Improving Human Health and Healthy Aging and Promoting Quality Life-A Review*, National Library of Medicine, Volume 65, pages 299-308, 2010, DOI: 10.1007/s11130-010-0177-1

[10] Vildana Alibabic, Azra Skender, Melisa Orascanin and Edina Sertovic, *Evaluation of Morphological, Chemical and Sensory Characteristics of Raspberry Cultivars Grown in Bosnia and Herzegovina*, Turkish Journal of Agriculture and Forestry 42, January 2017, DOI: 10.3906/tar-1702-59

[11] Sarmite Strautina, Inta Krasnova, I. Kalnina and Kaspars Kampuss, *Results of red raspberry breeding in Latvia*, Acta Horticulturae 946 (946):171-176, April 2012, DOI: 10.17660/ActaHortic.2012.946.26

[12] IndexBox, *Raspberries and Blackberries-Market Analysis, Forecast, Size, Trends an Insights*, February 1, 2024 Latvia, Source: https://www.indexbox.io/store/latvia-raspberries-and-blackberries-market-analysis-forecast-size-trends-and-insights/

[13] Katerina Bojkovska, Nikolce Jankulovski, Goran Mihajlovski and Jovica Momirceski, *Analysis o Market Opportunities for Raspberry production in the Republic of North Macedonia*, International Journal of Research, December 2020, Vol 8 (12), 149-154, DOI: https://doi.org/10.29121/granthaalayah.v8.i12.2020.2698

[14] Sapic Srdjan, Jaksic Milena and Stojkovic Dragan, *Berza maline u Srbiji-ispitivanje stavova proizvodjaca*, Ekonomika preduzeca, 2020, vol.68, br.3-4, str.215-228

[15] Strautina Sarmite, Kalnina Ieva, Kaufmane Edite, Sudars Kaspars, Namatevs Ivars, Nikulins Arturs, Judvaitis Janis and Balas Rihards, *YOLOv5 Deep Neural Network for Quince and Raspberry Detection on RGB Images*, Publisher Zenodo, DOI: 10.5281/zenodo.7014727

[16] Sarmite Strautina, Ieva Kalnin, Arturs Nikulins, Edite Kaufmane, Kaspars Sudars, Ivars Namatevs and Edgars Edelmers, *RaspberrySet: Dataset of Annotated Raspberry Images for Object Detection*, MDPI, 2023, Dataset: https://doi.org/10.5281/zenodo.7014728

[17] Achyut Morbekat, Ashi Parihar and Rashmi Jadhav, *Crop Disease Detection Using YOLO*, International Conference for Emerging Technology (INCET), 2020.

[18] Om Tiwari, Vidit Goyal, Pramod Kumar and Sonakshi Vij, *An experimental set up for utilizing convolutional neural network in automated weed detection*, 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU), 2019.

[19] Jiya Tian, Yong Zhang, Yanling Wang and Chao Wang, *A Method of Corn Disease Identification Based on Convolutional Neural Network*, 12th International Symposium on Computational Intelligence and Design (ISCID), 2019.

[20] T. Maleva, A. Kuznetsova, and V Soloviev, *Detecting Apples in Orchards Using YOLOv3 and YOLOv5 in General and Close-Up Images*, International Symposium on Neural Networks, 2020.

[21] Antonio Quintero Rincon, Marco Mora, Jose Naranjo-Torres, Claudioe Fredes and Andres Valenzuela, *Raspberries-LITRP Database: RGB Images Database for the Industrial Applications of Red Raspberries' Automatic Quality Estimation*, Appl.Sci. 2022, 12(22), 11586.

[22] Evan Juras, *TensorFlow Lite Object Detection Model Performance Comparison, EJ Technology Consultants*, https://www.ejtech.io/learn/tflite-object-detection-model-comparison.

[23] Feng Xiao, Haibin Wang, Yuegin Xu and Ruiging Zhang, *Fruit Detection and Recognition Based on Deep Learning for Automatic Harvesting: An Overview and Review*, Agronomy 2023, 13, 1625.

[24] Sarmite Strautina, Ieva Kalnina, , Edite Kaufmane, Kaspars Sudars, Ivars Namatevs, Arturs Nikulins and Edgars Edelmers, *RaspberrySet: Dataset of Annotated Raspberry Images for Object Detection*, https://zenodo.org/records/7014728.

[25] Sarmite Strautina, Ieva Kalnina, , Edite Kaufmane, Kaspars Sudars, Ivars Namatevs, Arturs Nikulins and Edgars Edelmers, *RaspberrySet: Dataset of Annotated Raspberry Images for Object Detection*, Data 2023, 8(5), 86, https://doi.org/10.3390/data8050086.

[26] Nur E Aznin Ahsan Mimma, Sumon Ahmed, Tahsin Rahman and Riasat Khan, *Fruits Classification and Detection Application Using Deep Learning*, Sci.Program. 2022, 419874 [CrossRef]

[27] O. Enrique Apolo, Jorge Martinez Guanter, Gregorio Egea and P. Raja, *Deep Learning techniques for estimation of the yield and size of citrus fruits using UAV*, Eur.J.Agron. 2020, 115, 126030 [CrossRef]

[28] Junhyung Kang, Shahroz Tariq, Han Oh and Simon S. Woo, *A Survey of Deep Learning-Based Object Detection Methods and Datasets for Overhead Imagery*, IEEE Access, February 25, 2022.

[29] YOLO: https://docs.ultralytics.com/

[30] RetinaNet: https://keras.io/examples/vision/retinanet/

[31] SSD: https://pytorch.org/hub/nvidia_deeplearningexamples_ssd/

[32] MobileNet: https://keras.io/api/applications/mobilenet/

[33] EfficientDet: https://catalog.ngc.nvidia.com/orgs/nvidia/resources/efficientdet_for_pytorch

[34] Faster R-CNN: https://pytorch.org/vision/master/models/faster_rcnn.html

[35] Mask R-CNN: https://pytorch.org/vision/stable/models/mask_rcnn.html

[36] Detectron2: https://ai.meta.com/tools/detectron2/

[37] SPP-Net: Kaiming He, Xiangyu Zhang, SHaoqing Ren and Jian Sun, *Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition*, IEEE, April 2015.

[38] Cascade R-CNN: https://paperswithcode.com/paper/cascade-r-cnn-delving-into-high-quality

[39] Joseph Redmon, Santosh Divvala, Ross Girshick and Ali Farhadi, You Only Look Once: Unified, Real-Time Object Detection, IEEE, June 2016, DOI: 10.1109/CVPR.2016.91.

[40] Vishnu Onkar, Onkar Kunte, *Stall Number Detecton of Cow Teats Key Frames*, ResearchGate, November 2023, DOI:10.13140/RG.2.10551.70566.

[41] Joseph Redmon, Santosh Divvala, Ross Girshick and Ali Farhadi, *You Only Look Once: Unified, Real-time object detection*, In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, pages 779-788, 2016.

[42] Joseph Redmon and Ali Farhadi, *Yolo9000: better, faster, stronger*, In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, pages 7263-727, 2017.

[43] Joseph Redmon and Ali Farhadi, *Yolov3: An incremental improvement*, arXiv preprint arXiv: 1804.02767, 2018.

[44] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao, *Yolov4: Optimal speed and accuracy of object detection*, arXiv: 2004.1093, 2020.

[45] Nelson Joseph and Solawetz Jacob, *Yolov5 is here: Stateof-the-art object detection at 140 fps*, https://blog.roboflow.com/yolov5-is-here/, accessed: 2023-03-15.

[46] Chuyi Li, Lulu Li, Hongliang Jiag, Kaiheg Weng, Yifei Geng, Liang Li, Zaidan Ke, Qingyuan Li, Meng Cheng, Weiqiang Nie, *Yolov6: A single-stage object detection framework for industrial applications*, arXiv:2209.02976, 2022.

[47] Chien-Yao Wang, Alexey Bochkovskiy and Hong-Yuan Mark Liao, *Yolov7: Trainable bag-of-freebies sets new state-of-the-at for real-time object detectors*, arXiv: 2207.02696, 2022.

[48] Haiying Liu, Fengqian Sun, Jason Gu and LIxia Deng, *SF-YOLOv5: A Lightweight Small Object Detection Algorithm Based on Improved Feature Fusion Mode*, MDPI, Sensors 2022, 22, 5817.

[49] Neubeck A., Van Gool L., *Efficient non-maximum suppression*, In Proceedings of the 18th International Conference on Pattern Recognition (ICPR'06), Hong Kong, China, 20-24 August 2006, Volume 3, pp. 850-855.

[50] Abhishek Kumar Pandey, *SiLU (Sigmoid Linear Unit) activation function*, Medium, April 2024, https://medium.com/@akp83540/silu-sigmoid-linear-unit-activation-function-d9b6845f0c81.

[51] Hyun-Ki Jung and Gi-Sang Choi, *Improved YOLOv5: Efficient Object Detection Using Drone Images under various conditions*, Appl. Sci. 2022, 12(14), 7255.

[52] Kewen Xia, Zhongliang Lv, Chuande Zhou and Guojun Gu, *Mixed Receptive Fields Augmented YOLO with multi-path spatial Pyramid pooling for steel surface defect detection*, ResearchGate, May 2023, Sensors 23(11):5114.

[53] Chien-Yao Wang, Hong-Yuan Mark Liao, Yueh-Hua Wu, Ping-Yang Chen, Jun-Wei Hsieh and I-Hau Yeh, *CSPNet: A New Backbone that can Enhance learing capability of CNN*, IEEE, July 2020, DOI:10.1109/CVPRW50498.2020.00203.

[54] Hyeokjin Park and Joonki Paik, *Pyramid Attention Upsampling Moule for Object Detection*, IEE Access, April 2022.

[55] Tanvir Parvez Matobbar, *Defective insulator detection and recognition based on deep convolutional neural network*, ResearchGate, May 2023, DOI:10.13140/RG.2.2.36311.65443

[56] *Facebook Research / MaskRCNN – Benchmark* , 18 December 2021, GitHub, https://github.com/facebookresearch/maskrcnn-benchmark

[57] Francisco Javier Yague, Jose Francisco Diez-Pastor, Pedro Latorre-Carmona and Cesar Ignacio Garcia Osorio, *Defect detection an segmentation in X-Ray images of magnesium alloy castings using the Detecton2 framework*, Departamento de Ingenieria Informatica, Universidad de Burgos. Avda. Cantabria s/n, Burgos, Spain, February 2022.

[58] Fatma Akalin and Nejat Yumusak, *Detecton and classification of white blood cells with an improved deep learning-based approach*, Turkish Journal of Electrical Engineering and Computer Sciences, Volume30, Number 7, Article 16.

[59] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother and Piotr Dollar, Panoptic Segmentation, *Proceedings of the IEEE/CVF Conference on ComputerVision and Pattern Recognition (CVPR)*, June 2019.

[60] Marc Ackermann, Sebastian Wesselmecking, Deniz Iren and Ulrch Krupp, *Automated segmentation of martensite-austenite islands in bainitic steel*, Materials Characterization, July 2022, ResearchGate, DOI: 10.1016/j.matchar.2022.112091.

[61] Grant Merz, Yichen Liu, Colin J. Burke, Patrick D. Aleo, Xin Liu, Matias Carrasco Kind, Volodymyr Kindratenko and Yufeng Liu, *Detection, instance segmentation and classification for stronomical surveys with deep learning (DEEPDISC): DETECTRON2 implementation and demonstration wit Hyper Suprm-Cam data*, Advance Access publication, September 2023.

[62] T. Akiba, S. Sano, T.Yanase, T. Ohta, M. Koyama, *Optuna*, Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, ACM, New York, NY, USA, 2019, pp.2623-2631, 07252019.

[63] Juan Terven, Diana-Margarita Cordova-Esparza and Julio-Alejandro Romero-Gonzalez, *A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS*, Mach.Learn.Knowl.Extr, 2023, 5(4), 1680-1716.

[64] Christine Dewi, Dhananjay Thiruvady and Nayyar Zaidi, *Fruit Classification System with Deep Learning and Neural Architecture Search*, ResearchGate, June 2024.

[65] Yuqiao Liu, Yanan Sun, Bing Xue, Mengjie Zhang, Gary G. Yen and Kay Chen Tan, *A Survey on Evolutionary Neural Architecture Search*, IEEE Transactions on Neural Networks and Learning Systems, 2021.

[66] Vasco Lopes, Fabio Maria Carlucci, Pedro m. Esperanca and Marco Singh, *Manas: Multi-agent neural architecture search*, Machine Learning 113(1):1-24., ResearchGate, September 2023.

[67] Daniel Dluznevskij, Pavel Stefanovic and Simona Ramanauskaite, *Investigation of YOLOv5 Efficiency in Iphone Supported Systems*, ResearchGate, January 2021, 9(3).

[68] Zuzana Reitermanova, *Data Splitting*, WDS'10 Proceedings of Contributed Papers, Part I, 31-36, 2010.

Biography

Simona Fimić was born on September 2, 1997, in Novi Sad, Serbia. She attented elementary school “Sonja Marinković” and grammar school “Laza Kostić“ in Novi Sad. In 2016, she started her bachelor’s degree studies in Financial Mathematics at the Faculty of Sciences, University of Novi Sad, which she completed in Septemebr 2021. The same year, she continued her education at the same university, enrolling in a master’s degree program in Applied Mathematics – Data Science. She completed the master’s program with a GPA of 8.12.