KTH ROYAL INSTITUTE
OF TECHNOLOGY

Degree Project in Computer Science

Second cycle, 30 credits

Drone Detection using Deep Learning

YATING LIU

Stockholm, Sweden, 2023

Drone Detection using Deep Learning

YATING LIU

Master's Programme, Computer Science, 120 credits
Date: February 8, 2023

Supervisor: Mårten Björkman
Examiner: Erik Fransén
School of Electrical Engineering and Computer Science
Host company: Skysense AB

Abstract

Drone intrusions have been reported more frequently these years as drones become more accessible in the market. The abuse of drones puts threats to public and individual safety and privacy. Traditional anti-drone systems use radio-frequency sensors widely to get the position of drones. In this thesis, deep-learning-based detection algorithms on surveillance cameras have been investigated to be integrated into the RF anti-drone system. The objective of the thesis is to evaluate state-of-the-art models and training strategies for drone detection. The main challenges in this thesis were detecting small drone targets at long distances and running the model in real-time. It is difficult to find a publicly available dataset of small drones online, so a real-world small drone dataset was constructed and used in this thesis. Different versions of YOLO were compared and tested on the real-world dataset. Modifications on the detection heads of the models were conducted to examine their effects on small object detection. The method of tiling on datasets was also adopted to help with the detection of small drones. Images from different sources were trained and added to compare with the model trained with only one source. Bird images were added to the training dataset in different ways for reducing the false positives when birds were included in the test set.

In conclusion, YOLOv5n and YOLOv5m overall yielded the best results in precision, recall, and inference speed. The additional detection head on shallow layers at small scales can improve the precision at the cost of recall and inference time. However, it was not effective when the objects were extremely small. The tiling approach yielded the most effective improvement in increasing the recall value of the model. Adding bird images into the training dataset as background had better precision and recall value than adding annotated birds as a separate class in training. Training without extremely small drones and birds helped improved the precision metric, while the recall value would decrease. Further study is required to examine the generality of the results in all types of drones.

Keywords

Drone Detection, Real-time Tracking, Deep Learning, Computer Vision, Small Object Detection, YOLO

Sammanfattning

Drönarintrång har rapporterats oftare dessa år allteftersom drönare blir mer tillgängliga på marknaden. Missbruket av drönare utgör ett hot mot allmänhetens och individens säkerhet och integritet. Traditionella antidronesystem använder radiofrekvenssensorer i stor utsträckning för att få reda på drönarnas position. I denna avhandling har djupinlärningsbaserade detektionsalgoritmer för övervakningskameror undersökts för att integreras i RF-antidrönarsystemet. Syftet med avhandlingen är att utvärdera state-of-the-art modell och träningsstrategier för drönardetektering. De största utmaningarna i denna avhandling var att upptäcka små drönarmål på långa avstånd och att exekvera modellen i realtid. En verklighetsbaserad mindre drönardatauppsättning konstruerades. Olika versioner av YOLO jämfördes och testades på den verkliga datamängden. Modifieringar av modellernas detektionshuvuden genomfördes för att undersöka deras effekter på detektering av små föremål. Metoden för så kallad tiling av datauppsättningar prövades också för att hjälpa till med upptäckten av små drönare. Bilder från olika källor tränades och lades till för att jämföra med modellen tränade med endast en källa. Fågelbilder lades till datasetet på olika sätt för att minska de falska positiva i de fall fåglar även fanns med i testdatamängden..

Sammanfattningsvis gav YOLOv5n och YOLOv5m totalt sett de bästa resultaten i precision, recall-värde och beräkningshastighet. Det extra detekteringshuvudet på grunda lager i små skalor kan förbättra precisionen på bekostnad av återkallelse och beräkningstid. Det var dock inte effektivt när föremålen var extremt små. Tiling-metoden gav den mest effektiva förbättringen för att öka recall-värdet hos av modellen. Att lägga till fågelbilder i träningsdatauppsättningen som bakgrund hade bättre precision och recall-värdet än att lägga till kommenterade fåglar som en separat träningsklass. Träning utan extremt små drönare och fåglar bidrog till att förbättra precisionsmåttet, medan recall-värdet skulle minska. Ytterligare studier krävs för att undersöka generella resultaten i alla typer av drönare.

Nyckelord

Drönardetektering, Realtidsspårning, Deep Learning, Computer Vision, Small Object Detection, YOLO

Acknowledgments

I would like to thank Mårten Björkman and Erik Fransén for giving academic guidance throughout the degree project. I would also like to thank Peng Wang for allowing me to conduct my degree project at Skysense AB. It was a great pleasure to work at Skysense for a few months with encouraging colleagues and other master thesis students. Finally, I would like to thank all my friends and my family for supporting me during my studies.

Shanghai, January 2023 Yating Liu

Stockholm, February 2023
Yating Liu

Contents

List of Figures

List of acronyms and abbreviations

FPN	Feature Pyramid Networks
HOG	Histogram of Oriented Gradients
PAN	Path Aggregation Network
PTZ	Pan-Tilt-Zoom
RCNN	Region Based Convolutional Neural Networks
RF	Radio Frequency
SAPP	Skysense Airspace Perimeter Protection
SOTA	state-of-the-art
SSD	Single Shot Detector
SURF	Speeded Up Robust Features
UAV	Unmanned Aerial Vehicle
YOLO	You Only Look Once

Chapter 1

Introduction

Unmanned Aerial Vehicles (UAVs), commonly known as drones, have become more accessible to customers, increasing the risks to the airspace of airports, prisons, power plants, and governments. Early this year in Sweden, two nuclear plants, the parliament, and the royal palace were intruded on by mysterious drones, raising the public's and the police's concerns about Sweden's preparedness against drones [1]. If the intruding drones are not detected and stopped at an early stage, the infrastructure may suffer from privacy leaks, aircraft collisions or other incidents. Traditional technologies used in drone tracking systems involve radar or radio-frequency detection, but these methods are not accurate when the drone is at a place with signal interference or the received signal is blocked. To verify the accuracy of the radio-frequency-based anti-UAV system and provide visual cues for security personnel, Skysense AB is seeking a way to integrate a Pan-Tilt-Zoom (PTZ) camera into the drone tracking system, where the camera can track the drone and show the real-time drone detection results with bounding boxes in video streams.

1.1 Background

This thesis will be carried out at Skysense AB, the leading company in anti-UAV technology in Sweden. Skysense currently possesses a drone surveillance system Skysense Airspace Perimeter Protection (SAPP) based on radio frequency, which can detect and tracks drones up to 3 kilometers away. So far, the system can output the position of the drone in radian angle form, while there is a lack of approach to show the visual information of the drone from a real-world perspective. To make the system more straightforward for

users, a PTZ camera needs to be integrated into SAPP that can automatically detect the drone according to the angle position from SAPP and move the camera viewpoint to keep the drone in the center of the frame. The scope of this thesis will focus on dataset construction and deep-learning model training for drone detection.

1.2 Problem

To meet the requirements of the camera system, the model should be fast enough to detect the drone in real time because drones can be fast-moving in frames. In order to have effective detection, the model is desired to detect drones when they are at distance at an early stage, so the drones in the image will be very small in pixels.

The research questions that will be investigated in this thesis include:

- What metrics are appropriate for evaluating drone detection performance?
- Which state-of-the-art (SOTA) model is best for real-time drone detection?
- How to improve the detection precision of small drones?
- How to reduce false positives caused by birds?

1.3 Purpose

Drone detection and surveillance have gained more and more attention as the usage of drones grows. The purpose of this thesis is to examine the performance of the SOTA models and different training strategies for drone detection. Companies or individuals who would like to implement drone detection on a camera may use this thesis as a guide for data collection, model selection, and training.

1.4 Ethics and Sustainability

The dataset used in the thesis contains both publicly available data and private data collected by Skysense AB, which will not be published publicly but can be included as examples in this thesis. The public dataset of drones is a synthetic

dataset that only contains drones and virtual backgrounds, so there is no human involved. The bird dataset and manually collected dataset by Skysense were also checked to ensure that no human was included. There is also no offensive or prejudiced label categories in these datasets. Since all the models are trained and tested on NVIDIA Agx Xavier, the experiment results may differ due to different hardware.

Another aspect regarding ethics in this thesis is to improve the anti-drone systems by adding visual aids, which can extend the target users from technicians to people with no professional knowledge of radar or radio frequency. With lower technical obstacles for users, more individuals and companies are able to protect critical infrastructure and privacy from drone intrusion. Infrastructures such as airports and nuclear plants don't need to be shut down or interrupted if they could stop drone intrusion in advance. Such intrusion prevention systems can save a lot of human resources as well as natural resources, obtaining sustainability.

1.5 Research Methodology

The methodology approaches in this thesis begin with the data collection by downloading publicly available drone datasets and manually creating drone datasets at Skysense AB. By doing a literature review, several SOTA models are selected as candidates for benchmarking. Then some modifications to the model and dataset are applied to see if they can improve the detection performance. Different training strategies are tested by training models from scratch with different training sets.

1.6 Delimitations

The final deliverable to Skysense AB will be a system that can receive the angle information from SAPP and center camera to the drone according to the detection result, but this thesis will not cover the camera control design or system design. The scope of the thesis is limited to object detection model performance evaluation and improvements. Since there is only one camera implemented at Skysense, the camera will only track one drone at a time if there are multiple detections. The system also does not need to identify whether the tracked target is the same instance. The drone type in this thesis only contains DJI Mavic because other types of drones are not in a good condition to fly at Skysense. All the images in the dataset are taken in daytime,

so the model and the system are not expected to work in the evening.

1.7 Structure of the thesis

Chapter 2 presents relevant background information about popular models on generic object detection and relevant literature on drone detection. Chapter 3 presents the methodology and method used to solve the problem, including the description of the dataset, evaluation metrics, the structure of the model, and experiment hardware and environment. Chapter 4 will show the results of the experiments. The results will be discussed in Chapter 5. Chapter 6 includes a conclusion about the thesis and a discussion about future work.

Chapter 2

Background

2.1 Drone Detection

The industrial and academic approaches to detect drone involve non-optical approaches using acoustics [2], radar [3] and radio frequency [4], and optical approaches which identify drones depending on extracted features from photos and videos [5] [6].

2.1.1 Non-Optical Approaches

Acoustics Detection The acoustic features of rotors of drones can be used to differentiate drones from surroundings [2]. Various sounds, including drones, airplanes, birds, and thunderstorms, are collected by microphone arrays and passed to algorithms to identify whether there is a drone based on high-frequency features [2]. However, this technique is not feasible in areas with much noise such as airports and urban [7].

Radar Detection Radar has been common in detecting large aerial vehicles like airplanes. Doppler radar is also modified to capture the frequency shifts of rotors and wing motions of drones in drone detection [3], while such modifications can introduce more noise and make it difficult to differentiate among drones, birds, and background clutter [8].

Radio Frequency detection Drones use Radio Frequency (RF) signals to communicate with the ground pilot and send videos to the controller. The RF signal can be captured and analyzed using method such as machine learning [9]. The detection of RF signals is effective and can be done at long distances, which makes it a common way of drone detection in the market [8].

2.1.2 Optical Approaches

Compared with the approaches presented previously, optical approaches are considered to be more convenient, intuitive, and economic. Optical approaches can be separated into two categories depending on how they extract the features for object detection and classification.

Approaches using handcrafted feature extraction The traditional approaches of object detection are based on handcrafted feature extractions and machine learning algorithms [10]. These approaches first use methods like background subtraction to find the area of interest [11]. In these applications, the majority of the cameras are quasi-static, which means the cameras move very slowly or do not move at all, and only targets move in the video streams [12]. Then feature descriptors such as Histogram of Oriented Gradients (HOG)) [11], Fourier descriptors [5], and Speeded Up Robust Features (SURF) [13] are extracted and sent to classifiers for classification and recognition. These methods usually take a shorter time but have a lower recognition rate compared to deep learning approaches.

Approaches using deep learning Deep learning in object detection has been extensively used nowadays. The feature extractions of deep learning methods are usually realized by convolutional layers, which are able to extract the higher-level semantic features of images besides the features from raw pixels. As a result, deep learning can generate better hierarchical features [10]. Many SOTA models have been tested and benchmarked for drone detection tasks [6] [14]. In this thesis, one of the goals is to test and compare some new SOTA objection models according to our drone detection requirements.

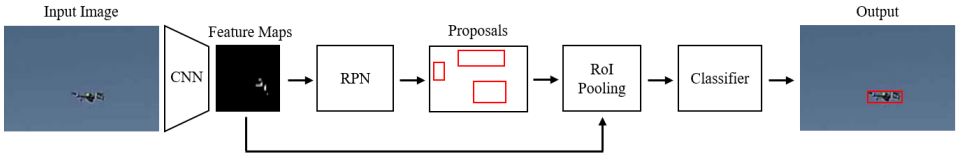
2.2 Deep learning in Object Detection

In the research area of computer vision, there are mainly three tasks: classification, detection, and segmentation. Our purpose of plotting a bounding box enclosing a drone in frames needs information about what and where is the object in the picture, falls in the detection category. There are two kinds of models for object detection in deep learning: two-stage detectors and one-stage detectors.

2.2.1 Two-stage Object Detection

The two-stage detectors are composed of two stages, where the first stage is used to generate a set of regions that may contain the target objects, and the

second stage classifies the proposed regions and produces bounding boxes. Typical two stage SOTA models are Region Based Convolutional Neural Networks (RCNN) [15], Fast RCNN [16] and Faster RCNN [17]. The structure of Faster RCNN is shown in Figure 2.1. The backbone of the model is a convolutional neural network, typically VGG-16 or ResNet, which produce feature maps. The Region Proposal Network generate proposals of locations where the objects would possibly occur. The RoI pooling takes the proposals and corresponding feature maps and do the max pooling to produce fixed size feature maps. The outputs from RoI pooling are then fed into the classifier to predict the class of the object and draw the bounding box. Two-stage detectors usually have higher accuracy than early one-stage detectors, while their inference speeds are too slow to do real-time detection.

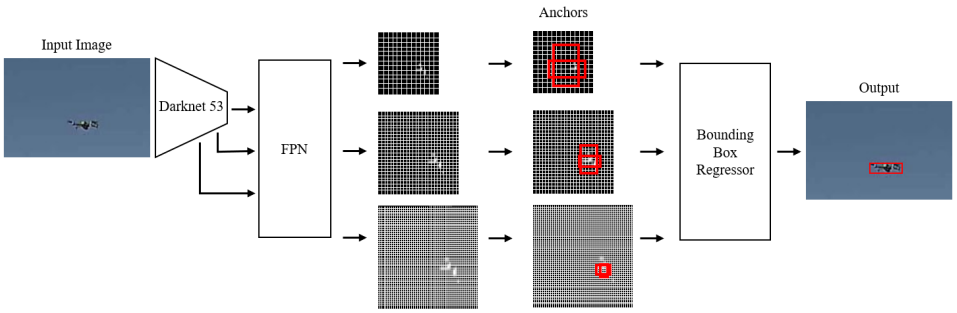


CNN: Convolutional Neural Network
RPN : Region Proposal Network
RoI : Regions of Interest

Figure 2.1: An example of two-stage object detection structure: Faster R-CNN

2.2.2 One-stage Object Detection

One-stage object detectors drop the region proposal phase of two-stage detectors and predict the bounding boxes and class of an object at the same time. Without losing much accuracy in detection, the one-stage detectors are fast enough to do real-time detection. The two popular one-stage detectors You Only Look Once (YOLO) [18] and Single Shot Detector (SSD) [19] take advantage of multi-scale feature maps from Feature Pyramid Networks (FPN) and anchor boxes to make predictions of the class of object and regression of bounding boxes on grid cells [6]. With further improvements in YOLO, the accuracy of the one-stage detector has been improved and become popular because of its good performance in both accuracy and inference speed. Figure 2.2 shows a classsic one-stage detector YOLOv3. Instead of generating proposals of objects' locations with neural networks, Yolov3 divides images into a grid of cells in three scales, and each cell predicts the anchor boxes and the class probabilities.

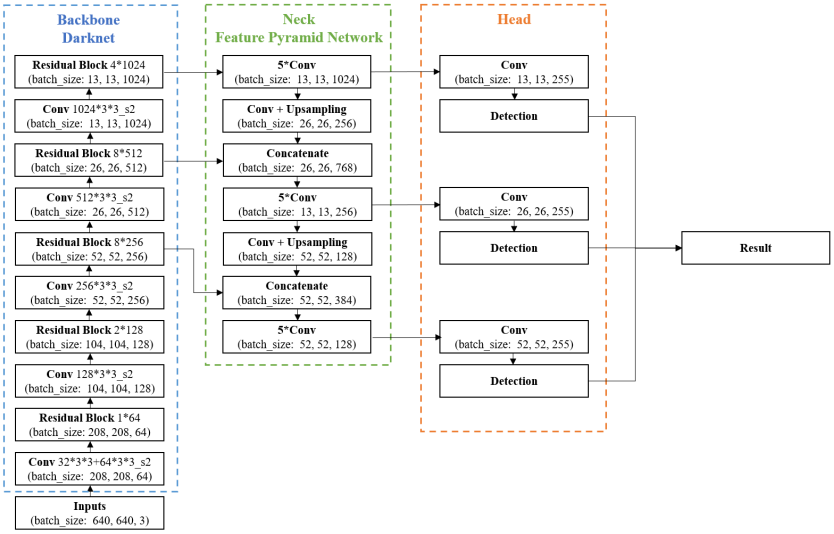


FPN: Feature Pyramid Networks

Figure 2.2: An example of one-stage object detection structure: YOLOv3

2.3 YOLO

In the past few years, the YOLO series algorithms have been updated to YOLOv5. The first three versions of YOLO were all proposed by the same author, and YOLOv3 was the one with big improvements in detection accuracy and inference speed [18]. YOLOv3 adopted Darknet53 as the backbone, utilized feature pyramid networks (FPN) as the neck. The backbone uses convolution layers to extract features from the input images, and the neck combines the features from previous convolution layers using up sampling [18]. The head also uses a pyramid structure, which make predictions of bounding boxes at three different scales (82nd layer, 94th layer and 106th layer) on the three feature maps generated from the neck [18]. The three output feature maps are of size 13×13, 26×26 and 52×52. The 13×13 layer is responsible for detecting large objects and 52×52 layer is responsible for detecting small objects. Each scale uses 3 pre-defined anchor boxes on each cell's center for predicting bounding boxes. In addition, skip connections are used to fuse the residuals between layers to improve the accuracy of detection [20]. The object class prediction and confidence are predicted using logistic regression [18].



Conv : convolutional layer(s)
_s2 : with stride of 2
Residual Block: repeated convolutional layers with ResNet structure
batch_size : the output size of the block

Figure 2.3: YOLOv3 Structure

In 2020, two new versions of YOLO, YOLOv4 and YOLOv5, have been released by two different research teams. Both models have shown improvement in performance. YOLOv4 changed the backbone to CSPDarknet53 for feature extraction, replaced the neck of YOLOv3 with the combination of FPN and Path Aggregation Network (PAN) and also explored alternatives in many other aspects of YOLOv3, including activations, bounding box loss, data augmentation, regularization methods, *etc.,* [21]. YOLOv5 has gained more popularity because besides trying to change the basic components of the networks, it has released different sizes of models and implemented the models in Python and Pytorch instead of C [22]. YOLOv5 provides models in five different scales: N, S, M, L, and X which stand for tiny, small, medium, large, and extra-large. The larger model has deeper networks and wider channels, meaning it has more layers and filters, while the structures of those five models are basically the same.

2.4 Small Object Detection

One of the challenges in object detection is detecting small objects. The definition of small objects here can be that they are either small in the real world or appears small in an image. The small object challenge in drone detection and tracking falls into the second category. A small object is defined as an object less than 32×32 pixels or the height and width occupy less than 10% of the width and height of the original image [14]. The small objects in images with low resolution usually lack distinctive shapes and textures, making detection more difficult. In recent years, many approaches have been proposed to improve small object detection.

2.4.1 Prediction pyramid

From YOLOv3, the models in YOLO family use a structure called prediction pyramid which combines the features extracted from both shallow and deep layers to make predictions. The models predict bounding boxes at three different scales on hierarchical feature maps [18]. In the deep layers of the network, the feature maps have larger receptive fields and rich semantic information, which is good for large object detection [23]. The feature maps produced from shallow layers contain small receptive fields, but the rich information of location and edges are preserved, which is better for small object detection [23].

2.4.2 Tiling

Object detection networks have different input sizes. For example, the input sizes of YOLOv3 and YOLOv4 are 416 $\times$ 416 pixels, while YOLOv5 takes 640 $\times$ 640 pixels as its input image sizes. When the input image is larger than the input sizes of the network, the image has to be resized to fit the network. For example, an image (1280 $\times$ 1280 pixels) with a drone (16 $\times$ 16 pixels) is passed to YOLOv5 as input, the image will then be resized to 640 $\times$ 640 pixels to fit the network. As a result, the drone will become 8 $\times$ 8 pixels. Moreover, the features of a small object would become extremely small or even vanish after it is operated by convolution layers and max-pooling layers, making it difficult to predict.

The method of tilling input images divides the image into patches and passes each patch to the network independently [24]. In this way, the operation of resizing the input image can be skipped and the detailed features can be

preserved since the resolution of the objects remains unchanged. However, the inference time will increase a lot because we need to feed every patch of the image into the network, and we also need extra time to crop the image and merge the predicted bounding boxes.

2.5 Related Works

A performance benchmark study on detecting drones has been done in [6]. It compares the mean average precision among Faster RCNN, SSD, YOLOv3 and DETR trained by three different datasets, and YOLOv3 overall yields the best performance in precision and inference speed [6]. In [25], the inference speed of YOLOv3 is improved by minimizing the number of filters while keeping the number of layers, and the detection of small drones is ensured with an additional camera with an external professional zoomed lens. It is not feasible in this thesis because there is only one camera available. Moreover, the performance of YOLOv4 and YOLOv5 are not included in these papers because YOLOv4 and YOLOv5 have not been released at that time. In [26], background subtraction is applied to the images before they are passed to YOLOv5s. However, the inference speed of the model is decreased and the background subtraction can only work on a fixed camera. Moreover, most of the drones in [26] are larger than 30 pixels $\times$ 30 pixels, which are of medium and large sizes. In the 2020 Drone vs. Bird Detection Challenge, two of the best three teams have tested YOLO in their work[27]. One team tested YOLOv5s and stopped investigating further because YOLOv5s did not give good detection results. The other team used YOLOv3 with Spatial Pyramid Pooling. They trained the model with real images and generated synthetic images with drones and birds, while the images are only annotated with drones. The synthetic images only improved the average precision by 0.2%. The team also applied tiling to the images, but the comparison of the model performance with and without the tiling strategy is not shown.

Chapter 3

Methods

The purpose of this chapter is to provide an overview of the research method used in this thesis. Section 3.1 focuses on the data collection construction for this research. Section 4.2 illustrates how an image was tiled. Section 3.3 describes the metrics used to evaluate how good a model is. Section 3.5 describes the experimental design. Section 3.6 describes the hardware types, software environments, and model hyperparameters.

3.1 Data Collection

The data used in the thesis were composed of real-world data and publicly available data. The real-world data was collected by myself at Skysense. The publicly available data was downloaded from the Internet. All the labels were converted to or created in YOLO format, which will be explained in 4.2.

3.1.1 Publicly available dataset

In this thesis, two datasets from online open resources were used. One was SimUAV [23], and the other one was Wild Bird dataset [28].

When searching the online drone dataset, it was found that few of the existing drone datasets contained drone targets in small sizes, or explicitly annotated different types of drones as different categories. The SimUAV dataset is a simulated drone dataset implemented by Airsim and UE4 [23]. Airsim is a drone simulator, and UE4 is used to create different environments for drones to fly in [23]. In this dataset, the author used four kinds of drone models including Parrot A.R. Drone, DJI Inspire I, DJI Mavic 2 Pro and DJI Phantom 4 Pro, and they were annotated with different labels. There were

4,095 images in SimUAV dataset that only contain DJI Mavic 2 Pro, which was used in this thesis to compare how the performance of the model would be if the training data is taken by a camera that is different from the test data. The images in this dataset were all 640 $\times$ 640 pixels.

Wild Bird dataset was collected at a wind farm in Japan with a digital still camera [28]. The resolution of the images was 5616 $\times$ 3744 pixels. There were 2,371 images that contained bird annotations in this dataset. The images were used to investigate whether adding bird images and annotations in the training set would improve the performance of the model in distinguishing drone and bird.

3.1.2 Real-world dataset

In this thesis, a realistic drone dataset was constructed with the help of Skysense and Molar Data. Videos of a DJI Mavic 2 Pro flying from 5 meters to 200 meters were taken using AXIS Q6215-LE (1920 $\times$ 1080 pixels) and iPhone (1920 $\times$ 1080 pixels and 4096 $\times$ 2160 pixels). Images were extracted from the video at the rate of 10 frames per second. There were 9,000 images in total (each camera had 3,000 images), and there was one and only one drone in each image. After the implementation of the system with the model trained by the above images, it was found that the model sometimes recognized birds as drones. Therefore, some additional images of birds were taken using AXIS Q6215-LE. Since controlling a bird or anticipating when and where they would appear, only 1,069 images of birds was taken, which was much fewer than the drone images.

3.2 Tiling

To boost the model performance on drone detection when the drone was small in the image, the method of image tiling was used. To prevent any miss of objects on the edge of the borders of tiled images, the cropped contiguous images need to have overlaps, and the length of overlaps should be longer than the width and height of the annotated bounding boxes in the dataset.

The resolution of the AXIS camera Q6215-LE was 1920 $\times$ 1080 pixels and the input image size of YOLOv5 was 640 $\times$ 640 pixels. In our thesis, each image in the dataset was cropped into 8 tiles, with 2 tiles in the vertical direction and 4 tiles in the horizontal direction. The width and height of the tile were both 640 pixels, which was the input image size of YOLOv5 model.

The formula for calculating the overlaps among evenly distributed tiles is

$$L_O = \frac{nL_{tile} - L_{image}}{n-1}.$$

where L_O is the length of the overlap, n is the number of tiles in that direction, L_{tile} is the length of a tile in that direction, L_{image} is the length of the original image in that direction.

According to this formula, the length of the overlap between the vertically contiguous tiles is 200 pixels, which is perfect because it is an integer. However, from this formula, the length of the overlap between the horizontally contiguous tiles is $213.\dot{3}$ pixels, such precision is difficult to achieve. Instead, the length of the three overlaps were manually designed to 200, 200, and 240 pixels to make the tiling more precise and easier to calculate. Figure 3.1 illustrates how an image is tiled. Figure 3.2 shows what the tiles of an image look like.

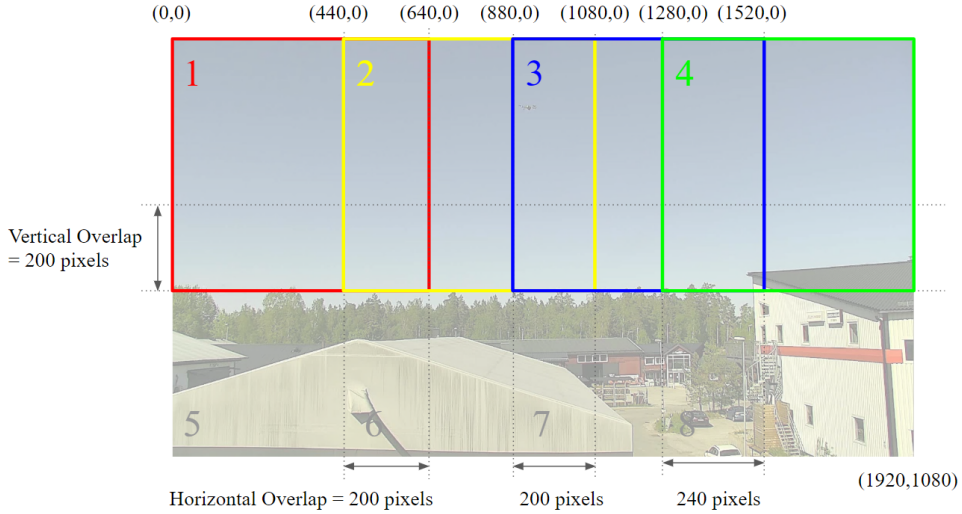


Figure 3.1: Tiling Strategy

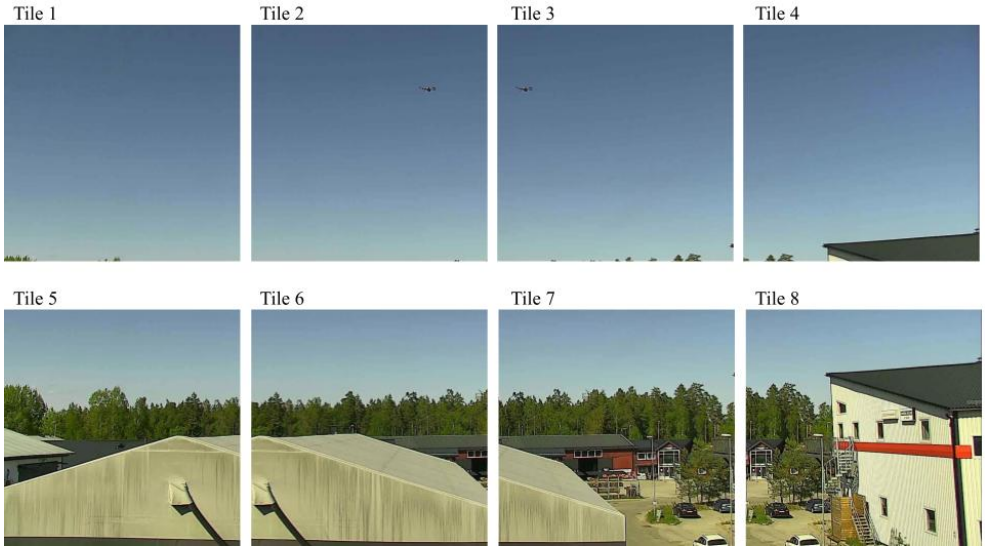


Figure 3.2: Eight tiles of a sample image

After tiling, the labels of the image also need to be adjusted to their new coordinates. All the labels of the original images are in YOLO format. For every image, there is a .txt file with the same name containing annotations of the objects, including the object class, x-y coordinates, width, and height. The last four values x, y, w, h are calculated in the equations below. The coordinate system uses pixel indices, where a pixel is a base unit. The origin lies at the top left corner of the image and the indices increase from left to right and top to bottom. The object class is an integer, and the rests are floats between 0 and 1, which are calculated as follows, and an example can be found in Figure 3.3. (x_{min}, y_{min}) is the coordinate of the top-left corner of the bounding box. (x_{max}, y_{max}) is the coordinate of the bottom-right corner of the bounding box. w_{image} is the width of the image and h_{image} is the length of the image.

$$x = \frac{x_{min} + x_{max}}{2} \frac{1}{w_{image}}$$

$$y = \frac{y_{min} + y_{max}}{2} \frac{1}{h_{image}}$$

$$w = \frac{x_{max} - x_{min}}{w_{image}}$$

$$h = \frac{y_{max} - y_{min}}{h_{image}}$$

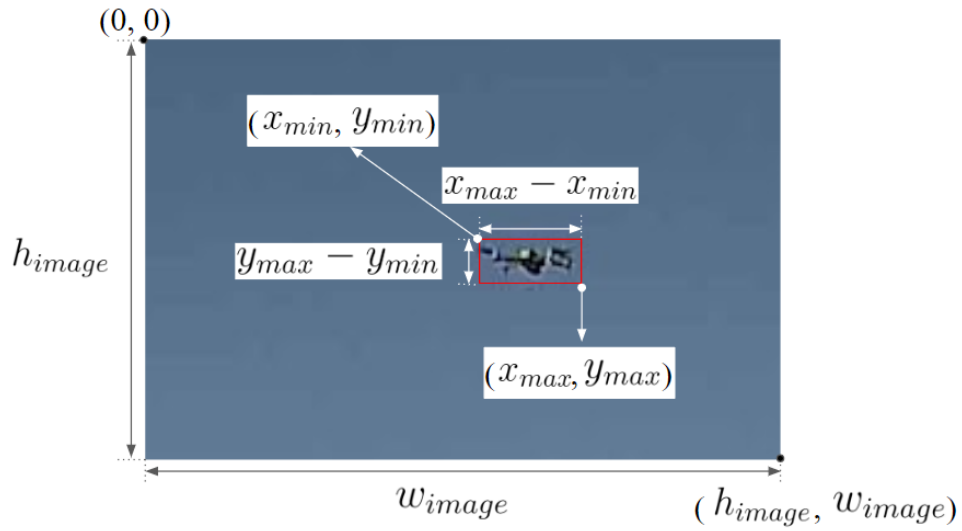


Figure 3.3: Example for YOLO format calculation

Since the x, y coordinates, the image height, and the image width all change in a new tile, we need to shift the annotations according to which tile they are in.

3.3 Evaluation Metrics

In this thesis, three evaluation metrics were used to evaluate the performance of models. They are inference time, precision, and recall.

3.3.1 Inference Time

The drone tracking system needed to be real-time, so the average inference time of models was calculated using a test dataset. To ensure a good user experience, the frames per second should be higher than 10, which means the inference time of the model should be less than 100 ms/frame. Moreover, the tiling increases the number of times to run the model. For each image, 8 tiles would be passed into the model, so to make the object detection fast enough for real-time tracking, the inference speed of the model should be less than 12.5 ms/tile.

3.3.2 Precision and Recall

The concept of Intersection of Union (IoU) was introduced to measure the performance of the object detection model. IoU is calculated as the intersection divided by the union of the two bounding boxes: ground truth bounding boxes and predicted bounding boxes (Figure 3.4). The ground truth bounding boxes are manually annotated and the predicted bounding boxes are the outputs of the model.

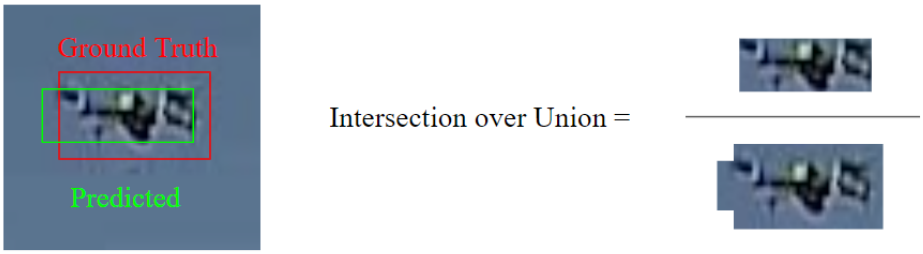


Figure 3.4: Example for calculating IoU

The higher the IoU, the more closely the predicted bounding box resembles the ground truth bounding box. A threshold is set to determine if the detected object is valid. In this thesis, the threshold is set to 0.65. In that case:

- If IoU $\geq$ 0.65, the detected object is valid and classified as True Positive (TP).
- If IoU $\leq$ 0.65, the detected object is not valid and classified as False Positive (FP).
- The case when the model fails to detect the ground truth in the image is classified as False Negative (FN).
- Every part that does not have ground truth and detected objects are classified as True Negative (TN).

Precision (P) and Recall (R) are calculated from true positives (TP), false positives (FP) and false negatives (FN).

$$P = \frac{TP}{TP + FP}$$

$$R = \frac{TP}{TP + FN}$$

Precision tells whether the detections are correct, and recall tells whether the detection can detect all the objects.

Ideally, we want both the precision and recall to be high, so the drone tracking system won't track a wrong object or miss a drone. In some cases, customers would not want any false alarms and may prefer a model with very high precision and do not care about the recall. Other customers that would like to track every suspicious target may want a model with high recall and allow reasonable false alarms.

3.4 Statistical Analysis

Friedman test and post hoc Nemenyi test were performed to compare the performance of different models in Python [29]. Each dataset was shuffled three times and recorded as D1, D2, and D3 for example. D1, D2, and D3 were then divided into the train, validation, and test sets. A model was trained and tested on D1, D2, and D3 separately and average values and standard deviations of the evaluation metrics were listed in this report. The results of each evaluation metric of a model belonged to one group. Afterward, the Friedman test was performed on the evaluation metrics of different models on each dataset. The Friedman test is used to validate significant differences among different models. The Friedman Test follows the hypothesis below:

- The null hypothesis (H_0): The mean value for the evaluation metric of the groups is equal.
- The alternative hypothesis (H_a): At least one of the evaluation groups' evaluation metrics differs from the others.

If the p-value of the Friedman test was larger than 0.05, the null hypothesis would be rejected. Then, a post hoc test, the Nemenyi test, would be conducted to find which two groups were different. The p values in this test described whether there was a statistical difference between the two groups. If the p-value of the evaluation metrics of the two groups was smaller than 0,05, the two models of the two groups had a statistically significant difference in the evaluation metric.

3.5 Experiments

3.5.1 SOTA model selection

In this section, two experiments were designed to find which YOLO model is best for detecting drones in real time. The first experiment was to compare the performance of YOLOv3, YOLOv4, and YOLOv5. The second experiment was to test whether adding or deleting the detection head would improve the performance of models.

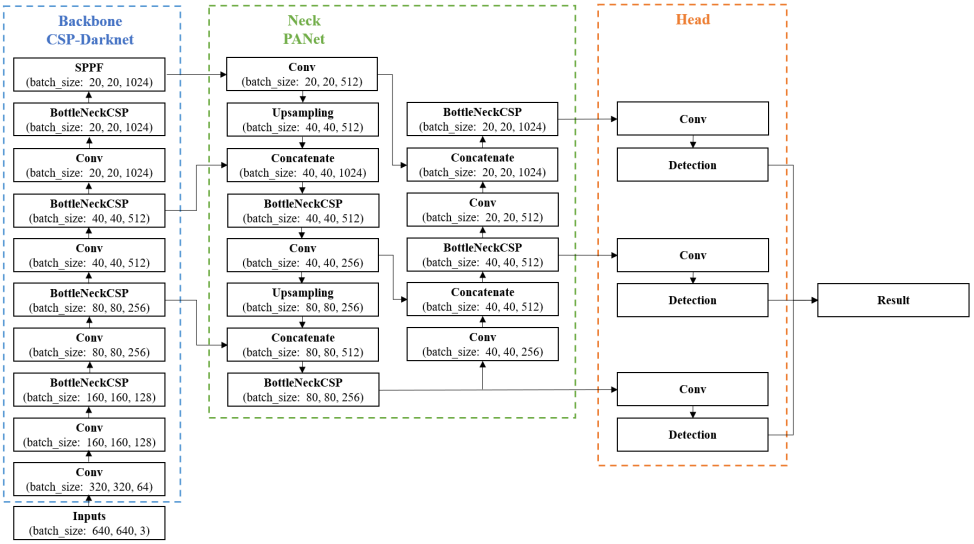
The 3,000 real-world images taken by AXIS are randomly divided into training, validation, and test datasets. The portion of the train, validation and test datasets is 8 : 1 : 1.

3.5.1.1 YOLO family benchmarking

YOLOv5 provides models in five different scales: N, S, M, L and X which stands for tiny, small, medium, large and extra-large. All five models together with YOLOv3 and YOLOv4 were trained from scratch for 300 epochs on GPU with batch size 16.

3.5.1.2 Impact of adding or deleting detection heads

Further modifications were applied to the best model selected from the previous experiment. From YOLOv3, YOLO family uses the prediction pyramid system as the head of the network to make the prediction of the bounding boxes. The system has three detection heads, gathering feature maps at three different scales [18].



Conv : convolutional layer(s)
CSP : Cross Stage Partial Network
SPPF : Spatial Pyramid Pooling-Fast
batch_size: the output size of the block

Figure 3.5: Structure of YOLOv5

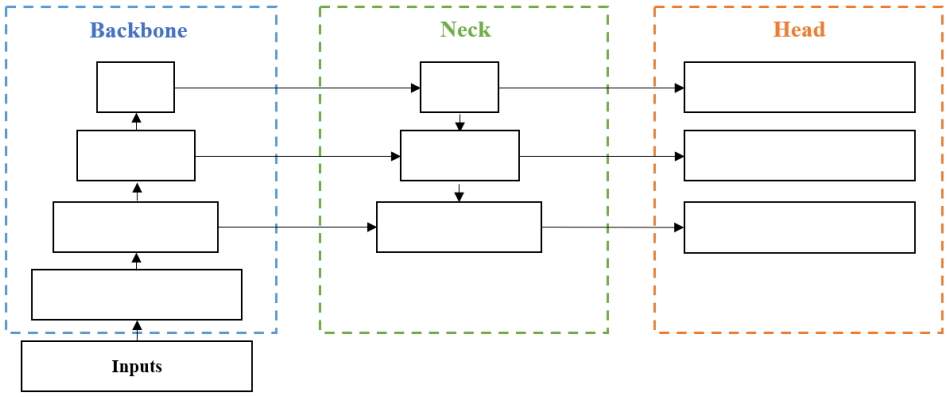


Figure 3.6: Simplified structure of YOLO

In this experiment, an additional detection head was added to the prediction pyramid system in a low-level layer to test whether it would improve the performance of detecting small objects. The detection head for detecting large

objects was also deleted to see if it would affect the small object detection performance.

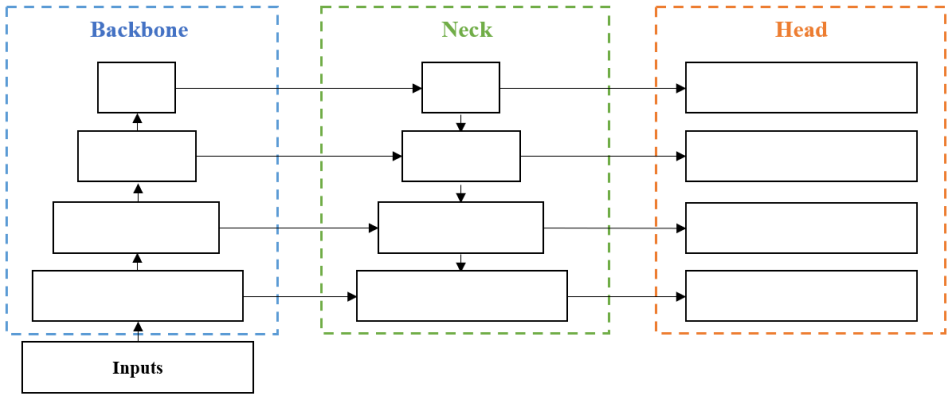


Figure 3.7: Simplified structure of YOLO with additional detection head

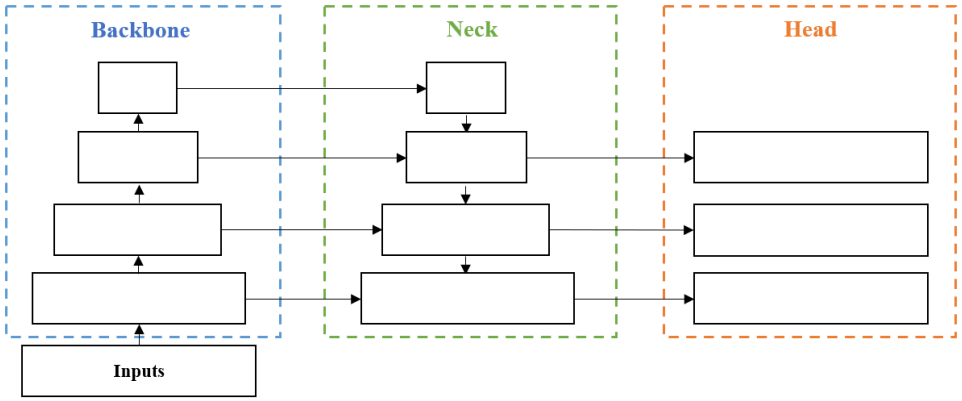


Figure 3.8: Simplified structure of YOLO with additional detection head and without detection head for large object

3.5.2 Impact of tiling

Every image from the train, validation and test dataset in previous experiments was tiled into 8 pieces, and the labels of drones were also recalculated for each tile. After tiling, 24,000 images were obtained. 4,247 images had a drone, while the rest were background. The experiment in this section compared the

performance of models trained on the original dataset and the tiled dataset. The impact of training without background was also tested.

3.5.3 Impact of camera

Skysense would like to offer more camera options for customers, so the camera type and resolution of images would not be restricted to AXIS Q6215-LE 1920×1080 pixels. In this experiment, how model performance would change was tested when it was trained on a dataset taken by a different camera, or a dataset that was simulated by a computer. The datasets used in this experiment include images taken by iphone in HD mode and 4K mode and simulated drone dataset SimUAV.

To investigate how many new images we need when we want to implement a model on the new camera, an additional experiment was done with models trained with merged datasets taken by phone in HD mode and AXIS.

3.5.4 Impact of adding bird images and annotations

When the tracking system with the model was tested at Skysense, the system would track birds sometimes. To reduce the false positives caused by birds, a bird dataset was constructed and trained together with images of drones. The bird dataset consisted of images from two sources. One was taken by AXIS at Skysense, which only had 1,069 images and 1,724 labels, while tiled AXIS dataset had 4,274 images and 4,274 labels of drones. To avoid the errors caused by imbalanced classes, the online bird dataset was added to the bird dataset which had 3,268 images and 3,843 labels. The model was trained with additional bird images with annotations as foreground and background. Then, images with extremely small drones were omitted from train and validation sets to test if it would reduce false positives. The YOLOv5 model with additional heads was also tested to see if the head would take effect without extremely small detection targets.

3.6 System Documentations

3.6.1 Hardware Specifications

In this thesis, NVIDIA Jetson AGX Xavier Developer Kit was used for all the development including data pre-processing, model training, and system

programming. The specifications of NVIDIA Jetson AGX Xavier Developer Kit are shown in the following Table 3.1.

Table 3.1: Hardware specifications of NVIDIA Jetson AGX Xavier

Components	Specifications
CPU	8-core ARM v8.2 64-bit CPU, 8MB L2 + 4MB L3
GPU	512-core Volta GPU with Tensor Cores
Memory	32GB 256-Bit LPDDR4x | 137GB/s

3.6.2 Environment

The models are developed and tested under the environments shown in the Table 3.2

Table 3.2: Environment Specifications

Name	Version
Ubuntu	18.04
CUDA	10.2
OpenCV	4.1.1
torch	1.8.0
torchvision	0.9.0
numpy	1.13
matplotlib	3.3.4
pandas	0.22.0
scipy	0.19.1

3.6.3 Hyperparameters

All the YOLO models in this thesis use the same hyperparameters setting.

Table 3.3: Hyperparameters Setting

Hyperparameters	Values	Comments
lr0	0.01	initial learning rate
lrf	0.01	final OneCycleLR learning rate (lr0 $\times$ lrf)
momentum	0.937	SGD momentum/Adam beta1
weight decay	0.0005	optimizer weight decay
warmup_epochs	3.0	warmup epochs (fractions ok)
warmup_momentum	0.8	warmup initial momentum
warmup_bias_lr	0.1	warmup initial bias lr
box	0.05	box loss gain
cls	0.5	cls loss gain
cls_pw	1.0	cls BCELoss positive_weight
obj	1.0	obj loss gain (scale with pixels)
obj_pw	1.0	obj BCELoss positive_weight
iou_t	0.20	IoU training threshold
anchor_t	4.0	anchor-multiple threshold
anchors	3	anchors per output layer (0 to ignore)
fl_gamma	0.0	focal loss gamma (efficientDet default gamma=1.5)
hsv_h	0.015	image HSV-Hue augmentation (fraction)
hsv_s	0.7	image HSV-Saturation augmentation (fraction)
hsv_v	0.4	image HSV-Value augmentation (fraction)
degrees	0.0	image rotation (+/- deg)
translate	0.1	image translation (+/- fraction)
scale	0.5	image scale (+/- gain)
shear	0.0	image shear (+/- deg)
perspective	0.0	image perspective (+/- fraction), range 0-0.001
flipud	0.0	image flip up-down (probability)
fliplr	0.5	image flip left-right (probability)
mosaic	1.0	image mosaic (probability)
mixup	0.0	image mixup (probability)
copy_paste	0.0	segment copy-paste (probability)

Chapter 4

Results and Analysis

4.1 SOTA model selection

4.1.1 YOLO family benchmarking

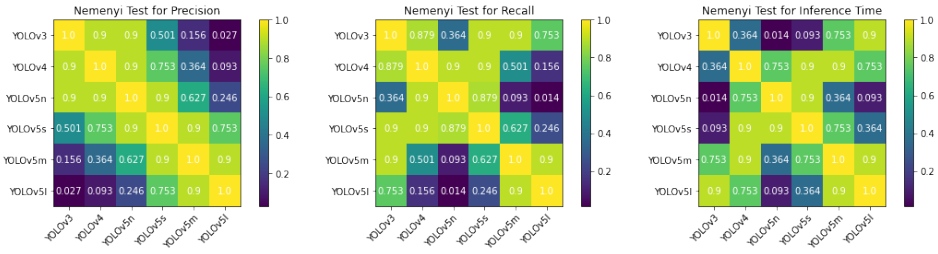
The different performances of YOLO models are shown in Table 4.1. The extra large version of YOLOv5 crashed every time while training, so there is no result of YOLOv5x. In YOLOv5 model collections, the larger the model is, the better precision and recall it has. In all the valid results we have, YOLOv5l has the best precision and recall but it is comparably slow, making it unsuitable for tiling methods. The YOLOv5n model is the fastest model and also the only model that can satisfy the time limit of tiling methods.

Table 4.1: Test results of YOLO family

Model	Precision (%)	Recall (%)	Inference time (ms)
YOLOv3	93.1 ± 0.6	75.1 ± 0.4	102.1 ± 0.2
YOLOv4	93.3 ± 0.8	74.1 ± 0.6	25.0 ± 0.3
YOLOv5n	93.8 ± 0.5	72.6 ± 0.6	$\mathbf{7.9 \pm 0.2}$
YOLOv5s	95.4 ± 0.5	74.1 ± 0.8	19.4 ± 0.3
YOLOv5m	96.1 ± 0.3	75.6 ± 0.6	49.5 ± 0.2
YOLOv5l	$\mathbf{98.8 \pm 0.3}$	$\mathbf{76.4 \pm 0.2}$	101.4 ± 0.3
YOLOv5x	N/A	N/A	N/A
p-value of the Friedman test	0.015	0.012	0.010

The p-values of Friedman tests on precision, recall and inference time were all smaller than 0.05, so the precision, recall and inference time had

statistically significant differences. Nemenyi test was done to find which models had different mean of precision, recall and inference time. The results were visualized in Figure 4.1.



(a) (b) (c)

Figure 4.1: Nemenyi Test for test results of YOLO family

In Figure 4.1a, only YOLOv3 and YOLOv5l had p-value less than 0.05, so only YOLOv3 and YOLOv5l had significantly different means of precision. For recalls, YOLOv5l and YOLOv5n had p-value $0.014 < 0.05$, so the recall of YOLOv5l was significantly greater than YOLOv5n. Although the average inference time of YOLOv5n was very short compared to other models, it was only significantly smaller than YOLOv3 according to the Nemenyi test.

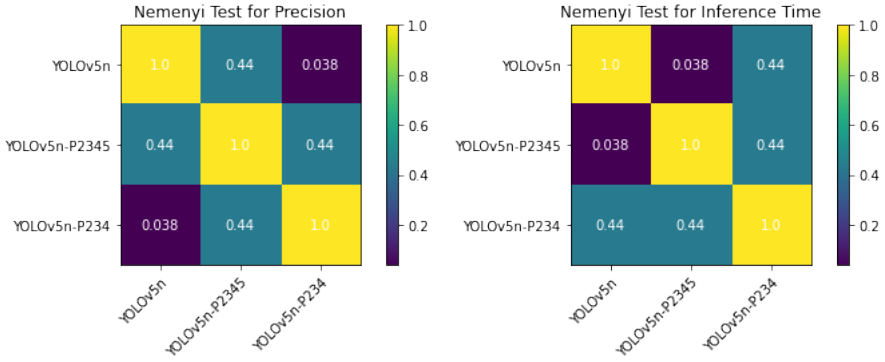
4.1.2 Impact of adding or deleting detection heads

Contrary to expectation, adding a detection head that took advantage of the feature maps with higher resolution had a negative impact on the performance of the model in all evaluation metrics. By deleting the detection head for large objects, the inference speed and recall became better but were still worse than the original model. The results are shown in Table 4.2. YOLOv5n-P2345 is the model with the additional detection head and YOLOv5n-P234 is the model without the detection head for large objects.

Table 4.2: Test results of YOLOv5n model and its variations

Model	Precision (%)	Recall (%)	Inference time (ms)
YOLOv5n	**93.8 ± 0.5**	**72.6 ± 0.6**	**7.9 ± 0.2**
YOLOv5n-P2345	93.4 ± 0.7	71.2 ± 0.4	10.5 ± 0.3
YOLOv5n-P234	90.0 ± 0.8	71.0 ± 0.8	9.9 ± 0.3
p-value of the Friedman test	0.049	0.086	0.049

The p-values of Friedman tests on precision and inference time were smaller than 0.05, while the recalls did not have statistically significant difference. According to the post hoc Nemenyi tests in Figure 4.2, the precision of YOLOv5n-P234 was significantly lower than YOLOv5n, and the inference time of YOLOv5n-P2345 was significantly longer than YOLOv5n. The results from the Friedman and Nemenyi tests both indicated that adding or deleting detection heads of YOLOv5n would not improve the performance of YOLOv5n.



(a) (b)

Figure 4.2: Nemenyi Test for test results of YOLOv5n model and its variations

4.2 Impact of tiling

The original AXIS dataset had 3,000 images and 3,000 labels. After tiling, the number of images in the tiled dataset was 8 times the number of images in the original dataset. The number of annotations was also increased because some drones appeared in overlapping areas. There were 24,000 images in the tiled AXIS dataset. 19,726 images in the tiled AXIS dataset were background, and the rest 4,274 images were labeled with drones. By comparing the position of labels in Figure 4.3 and Figure 4.4, drones became appearing on the corners of the images instead of only appearing in the center of the images. Drones were distributed more uniformly in images after tiling. The scale of the coordinates of the relative size became larger in the right picture in Figure 4.4, meaning that the same object took more portion of the image after tiling. Thus, the relative object sizes were larger.

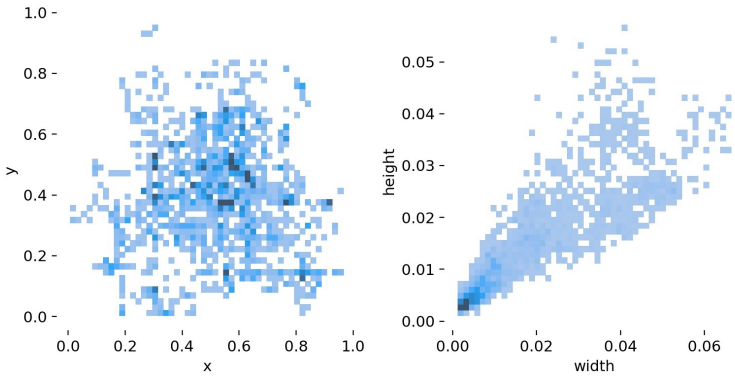


Figure 4.3: Data visualization of AXIS dataset without tiling: Position of labels (left) and Relative object size (right)

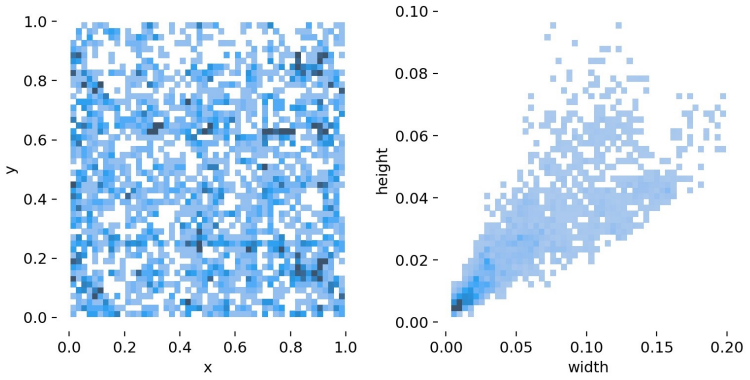


Figure 4.4: Data visualization of tiled AXIS dataset with tiling: Position of labels (left) and Relative object size (right)

YOLOv5n model was trained and validated with original and tiled images with and without background images. The results are shown in Table 4.4 below.

Table 4.3: Abbreviations of train, val and test sets 1

Abbreviation	Train	Val	Test
OOO	Original images	Original images	Original images
OOT	Original images	Original images	Tiled images
TTT	Tiled images	Tiled images	Tiled images
ttT	Tiled images w/o bg	Tiled images w/o bg	Tiled images

Table 4.4: Test results of training on original or tiled images

Abbreviation	Precision (%)	Recall (%)
OOO	**93.8 ± 0.5**	72.6 ± 0.6
OOT	72.9 ± 0.8	72.4 ± 0.6
TTT	93.3 ± 0.2	96.0 ± 0.3
ttT	92.6 ± 0.3	**96.4 ± 0.4**
p-value of the Friedman test	0.032	0.060

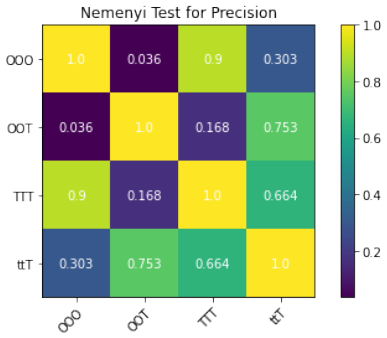


Figure 4.5: Nemenyi Test for test results of training on original or tiled images

The recall increased a little, while the precision became lower when the model trained with original images was tested with tiled images. That was because due to the tiling, the relative object size became larger, so what the model learned on original images was not applicable to tiled images, introducing more false positives. An example was shown in Figure 4.6.

When all the tiled images were used for training, the recall increased a lot. With a larger relative size in an image, the drone was easier to be detected by the model. However, the precision decreased by 0.01, which means the model mistook something else for a drone more often.

The performance of the model did not decrease a lot when trained without 19,726 images of background. It even had higher recall than trained with the background, which meant it has fewer misses on drones. An example was shown in Figure 4.7. In the following experiments, the pure background images in datasets would be removed to save space and time.

The Friedman and Nemenyi tests also verified the above analysis. The p-value of the Friedman test on recall was $0.060 > 005$. Thus, the null hypothesis that all the trained models had equal means of recall cannot be rejected.

For precision, only model trained on original size images had statistically significant difference between test set of original size images and tiled images.

(a) Original image where there is a drone in the middle. The model trained with original size images failed to detect the drone.

(b) Tiled image from the original image. The model trained with original size images detected false positive.

(c) Tiled image from the original image. The model trained with original-size images succeeded in detecting the drone.

Figure 4.6: Precision decreased and recall increased when the model trained with original size images was tested with tiled images

(a) Original image where there is a drone on the edge of the roof in the middle. The model trained with original-size images failed to detect the drone.

(b) Tiled image from the original image. The model trained with tiled-size images detected a false negative (top-left), while it also introduced a false positive (bottom-right).

Figure 4.7: Precision decreased and recall increased when the model trained with tiled size images was tested with tiled images

4.3 Impact of camera

Table 4.5 below shows how datasets were split into train, validation, and test sets. The test set was constructed from the tiled AXIS dataset only.

Table 4.5: Train, validation and test sets split for different dataset

Dataset	Train	Val	Test
Tiled AXIS	3420	427	427
Tiled PhoneHD	3040	381	/
Tiled Phone4K	2946	368	/
SimUAV	4095	512	/

The label distribution and relative sizes are shown in Figure 4.8, Figure 4.9, Figure 4.10 and Figure 4.11. The labels were distributed randomly and evenly in images in all datasets. The relative label size of tiled AXIS and tiled PhoneHD dataset were similar, where most of the labels were within the size of 16×6.4 pixels. Compared with tiled AXIS dataset, the tiled Phone4k dataset had larger relative label size and the SimUAV has smaller relative label size.

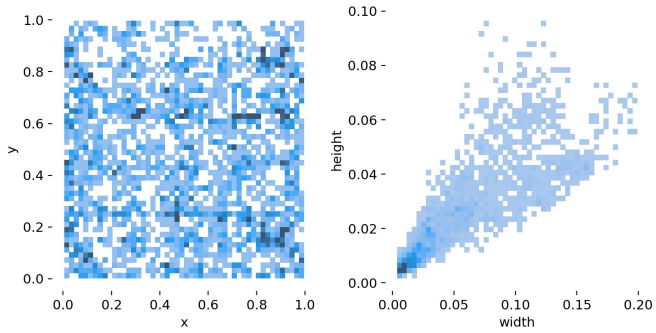


Figure 4.8: Data visualization of tiled AXIS dataset: Position of labels (left) and Relative object size (right)

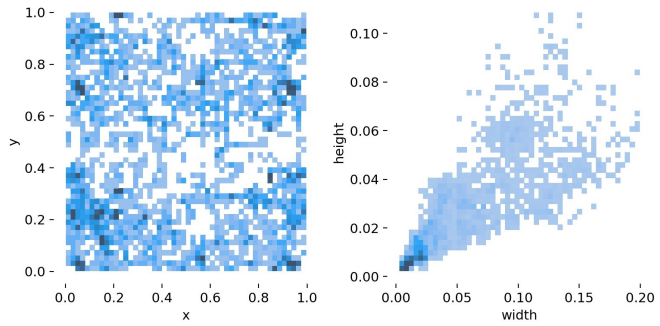


Figure 4.9: Data visualization of tiled PhoneHD dataset: Position of labels (left) and Relative object size (right)

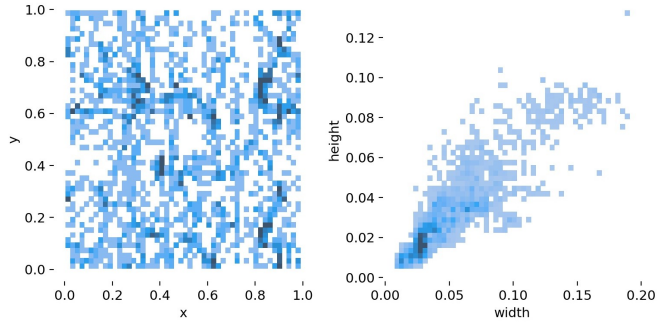


Figure 4.10: Data visualization of tiled Phone4K dataset: Position of labels (left) and Relative object size (right)

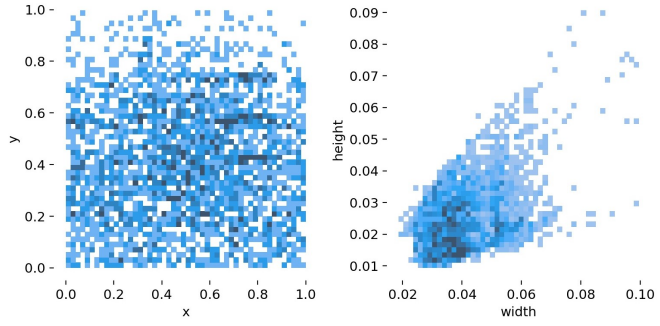


Figure 4.11: Data visualization of tiled SimUAV dataset: Position of labels (left) and Relative object size (right)

(a) A sample from tiled AXIS dataset

(b) A sample from tiled PhoneHD dataset

(c) A sample from tiled Phone4K dataset

(d) A sample from SimUAV dataset

Figure 4.12: Sample images from four datasets

Table 4.7 shows the results of training on images taken by different cameras. The SimUAV had the worst performance on the test set. It could be because the features of simulated Mavics were different from features of real Mavics and the sizes of the simulated Mavics were too small. So the model could not use the parameters learned from simulated Mavics to recognize real Mavics. The model trained with tiled PhoneHD had better performance but was still not comparable to the model trained with tiled AXIS. That could be because the relative label sizes were similar to the ones in tiled AXIS, but different cameras could have different brightness, colors, or distortion levels.

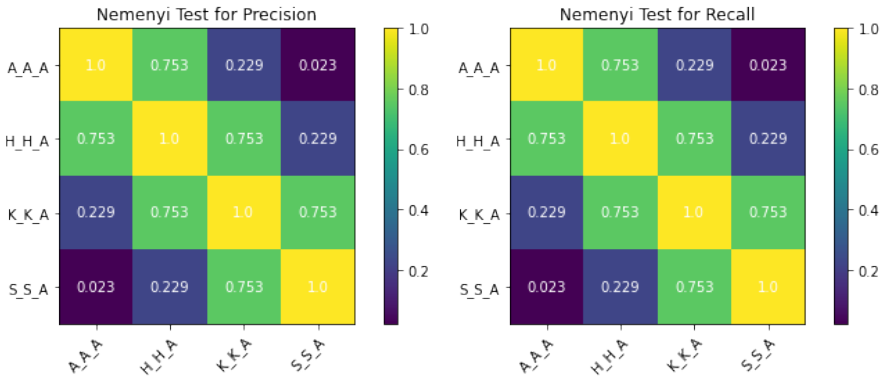
Table 4.6: Abbreviations of train, val and test sets 2

Abbreviation	Train	Val	Test
AAA	Tiled AXIS	Tiled AXIS	Tiled AXIS
HHA	Tiled PhoneHD	Tiled PhoneHD	Tiled AXIS
KKA	Tiled Phone4K	Tiled Phone4K	Tiled AXIS
SSA	SimUAV	SimUAV	Tiled AXIS

Table 4.7: Test results of training on images taken by different cameras

Abbreviation	Precision (%)	Recall (%)
AAA	**92.6 ± 0.3**	**96.4 ± 0.4**
HHA	82.0 ± 0.5	77.7 ± 0.4
KKA	47.0 ± 0.7	55.2 ± 0.3
SSA	9.5 ± 0.2	7.9 ± 0.7
p-value of the Friedman test	0.030	0.030

According to the Friedman and Nemenyi tests, the performance of the model trained on simulated dataset was significantly poorer than the model trained on tiled AXIS. The results above also show that to train a model that has the best performance on the test set, we should use the same camera to take photos for train, validation, and test sets. In the next experiment, the tiled phoneHD dataset and tiled AXIS were merged to investigate how many new images we need when we want to implement a model on the new camera. The results are shown in Table 4.8.



(a) (b)

Figure 4.13: Nemenyi Test for test results of training on images taken by different cameras

Table 4.8: Test results of merging Tiled PhoneHD and Tild AXIS

Dataset			Precision	Recall
Train	Val	Test		
Tiled PhoneHD	Tiled PhoneHD	Tiled PhoneHD	0.911	0.962
Tiled PhoneHD	Tiled PhoneHD	Tiled AXIS	0.825	0.776
Tiled AXIS	Tiled AXIS	Tiled AXIS	**0.923**	0.967
Tiled PhoneHD + 20% Tiled AXIS	Tiled PhoneHD + 20% Tiled AXIS	Tiled AXIS	0.836	0.858
Tiled PhoneHD + 40% Tiled AXIS	Tiled PhoneHD + 40% Tiled AXIS	Tiled AXIS	0.859	0.891
Tiled PhoneHD + 60% Tiled AXIS	Tiled PhoneHD + 60% Tiled AXIS	Tiled AXIS	0.905	0.907
Tiled PhoneHD + 80% Tiled AXIS	Tiled PhoneHD + 80% Tiled AXIS	Tiled AXIS	0.888	0.976
Tiled PhoneHD + 100% Tiled AXIS	Tiled PhoneHD + 100% Tiled AXIS	Tiled AXIS	0.891	**0.978**

With more images from tiled AXIS added to tiled phoneHD train dataset, the model performance became better. When the model was trained by all the train images from tiled AXIS and tiled phoneHD, the precision was still lower than the model trained only with tiled AXIS, but the recall was 0.11 higher. It indicates that the model was able to learn more features of Mavics

from different datasets, so the model had fewer false negatives. However, the decrease of precision also shows that the new feature learned from tiled phoneHD dataset may be misleading.

4.4 Impact of adding bird images and annotations

In this experiment, images of birds were added to the train, validation and test sets of tiled AXIS dataset. The label distribution and relative sizes of birds are shown in Figure 4.14.

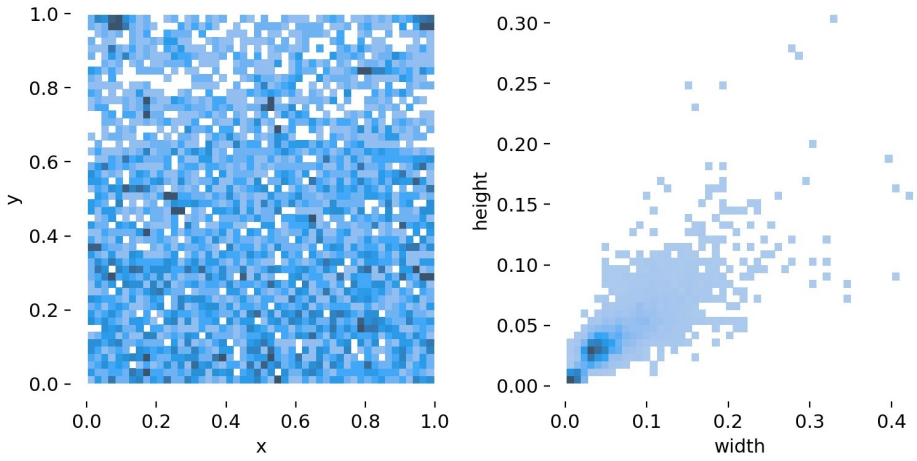


Figure 4.14: Data visualization of bird dataset: Position of labels (left) and Relative object size (right)

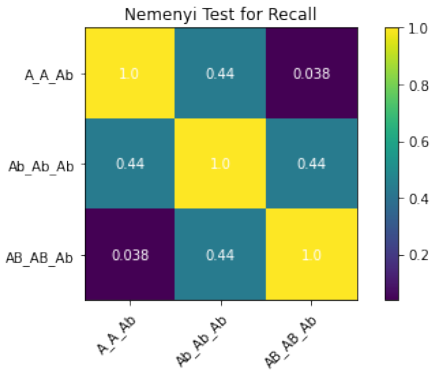
Table 4.10 shows that, the performance of model became poorer when birds were added to the test set. The Friedman and Nemenyi tests also shows that the recall of the model training with birds as foreground was significantly lower than training only on drones. The model performance would not be improved no matter how the bird dataset was added to the train and validation sets.

Table 4.9: Abbreviations of train, val and test sets 3

Abbreviation	Train	Val	Test
A_A_Ab	Tiled AXIS	Tiled AXIS	Tiled AXIS + birds
Ab_Ab_Ab	Tiled AXIS + birds as background	Tiled AXIS + birds as background	Tiled AXIS + birds
AB_AB_Ab	Tiled AXIS + birds as foreground	Tiled AXIS + birds as foreground	Tiled AXIS + birds

Table 4.10: Test results of training with bird images and labels

Abbreviation	Precision of drone (%)	Recall of drone (%)
A_A_Ab	**77.1 ± 0.2**	**96.4 ± 0.4**
Ab_Ab_Ab	77.0 ± 0.4	83.0 ± 0.4
AB_AB_Ab	76.1 ± 0.3	82.0 ± 0.5
p-value of the Friedman test	0.097	0.049



(a)

Figure 4.15: Nemenyi Test for test results of training with bird images and labels

To further investigated the impact of the bird dataset, the images with extremely small birds and drones and the corresponding labels were omitted from the train, validation, and test sets. The model was tested with both a complete test set and a test set without extremely small birds and drones. Here,

images in which a label's (width + height)/2 is smaller than 8 were omitted. The results were shown in Table 4.12. The overall performance of models trained with drones and birds was better than the model trained without birds. The model trained without birds mistook birds for drones many times. Adding birds in the train and validation set helped the model learn the difference between drones and birds. The test results of the third, forth, seventh, and eighth models in Table 4.12 also show that there is no need to label birds as a separate class.

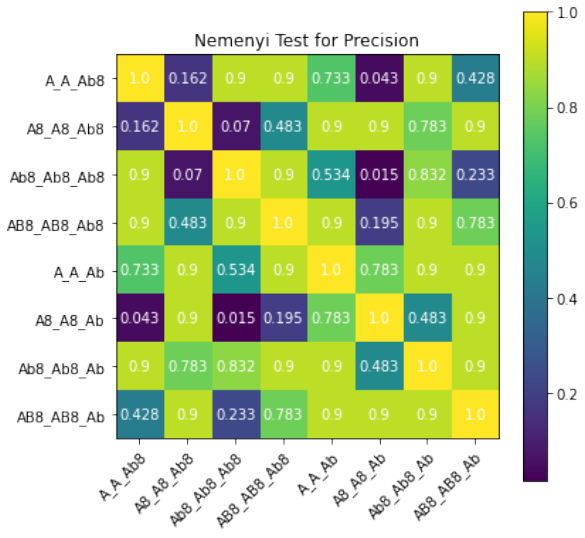
Table 4.11: Abbreviations of train, val and test sets 4

Abbreviation	Train	Val	Test
A_A_Ab8	Tiled AXIS	Tiled AXIS	Tiled AXIS + birds (-8)
A8_A8_Ab8	Tiled AXIS (-8)	Tiled AXIS (-8)	Tiled AXIS + birds (-8)
Ab8_Ab8_Ab8	Tiled AXIS + birds as background (-8)	Tiled AXIS + birds as background (-8)	Tiled AXIS + birds (-8)
AB8_AB8_Ab8	Tiled AXIS + birds as foreground (-8)	Tiled AXIS + birds as foreground (-8)	Tiled AXIS + birds (-8)
A_A_Ab	Tiled AXIS	Tiled AXIS	Tiled AXIS + birds
A8_A8_Ab	Tiled AXIS (-8)	Tiled AXIS (-8)	Tiled AXIS + birds
Ab8_Ab8_Ab	Tiled AXIS + birds as background (-8)	Tiled AXIS + birds as background (-8)	Tiled AXIS + birds
AB8_AB8_Ab	Tiled AXIS + birds as foreground (-8)	Tiled AXIS + birds as foreground (-8)	Tiled AXIS + birds

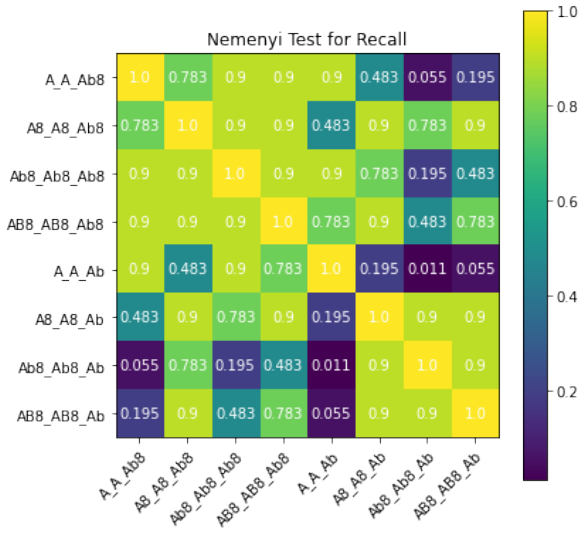
Table 4.12: Test results of omitting images with extremely small drones and birds

Abbreviation	Precision of drone (%)	Recall of drone (%)
A_A_Ab8	86.0 ± 0.1	**93.2 ± 0.1**
A8_A8_Ab8	46.0 ± 0.7	80.8 ± 0.4
Ab8_Ab8_Ab8	**86.3 ± 0.3**	92.5 ± 0.5
AB8_AB8_Ab8	84.0 ± 0.4	89.0 ± 0.6
A_A_Ab	77.1 ± 0.2	**96.4 ± 0.4**
A8_A8_Ab	32.1 ± 0.3	76.0 ± 0.2
Ab8_Ab8_Ab	**78.5 ± 0.3**	71.6 ± 0.5
AB8_AB8_Ab	72.3 ± 0.7	72.1 ± 0.6
p-value of the Friedman test	0.004	0.004

The results from the above two tables show that the model can learn the difference between birds and drones when labels in the train and validation images are bigger. According to the Friedman and Nemenyi tests, the precision of A8_A8_Ab was significantly lower than A_A_Ab8 and Ab8_Ab8_Ab8. Testing on images with bigger targets prevented the false positives in extremely small objects, ensuring good precision. The recall of A_A_Ab was significantly higher than Ab8_Ab8_Ab. The introducing of bird and the omitting small objects in training resulted in more false negatives. The following experiments in Table 4.13 tried to omit different sizes of small objects from the dataset. The images of birds were trained as background in these experiments.



(a)



(b)

Figure 4.16: Nemenyi Test for test results of omitting images with extremely small drones and birds

(a) Model trained with tiled AXIS + birds as background

(b) Model trained with tiled AXIS + birds as background (-8)

(c) Model trained with tiled AXIS + birds as background (-16)

(d) Model trained with tiled AXIS + birds as background (-32)

Figure 4.17: Different detection results when removing images with small objects. The object in the image is a bird.

(a) Model trained with tiled AXIS + birds as background

(b) Model trained with tiled AXIS + birds as background (-8)

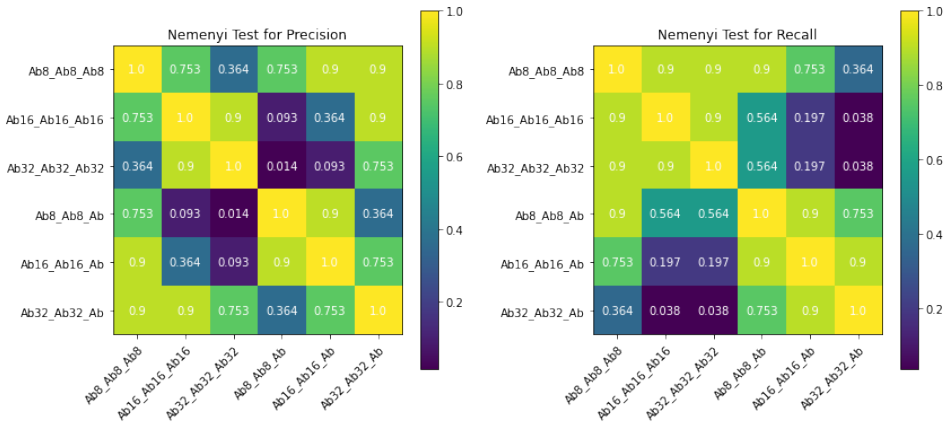
(c) Model trained with tiled AXIS + birds as background (-16)

(d) Model trained with tiled AXIS + birds as background (-32)

Figure 4.18: Different detection results when removing images with small objects. The object in the image is a drone.

Table 4.13: Test results of omitting more images with small drones and birds

Abbreviation	Precision of drone (%)	Recall of drone (%)
Ab8_Ab8_Ab8	86.3 ± 0.3	92.5 ± 0.5
Ab16_Ab16_Ab16	95.2 ± 0.3	$\mathbf{100.0 \pm 0.0}$
Ab32_Ab32_Ab32	$\mathbf{99.7 \pm 0.3}$	$\mathbf{100.0 \pm 0.0}$
Ab8_Ab8_Ab	78.5 ± 0.3	$\mathbf{71.6 \pm 0.5}$
Ab16_Ab16_Ab	84.8 ± 0.2	64.0 ± 0.6
Ab32_Ab32_Ab	$\mathbf{93.5 \pm 0.2}$	56.6 ± 0.2
p-value of the Friedman test	0.010	0.010



(a) (b)

Figure 4.19: Nemenyi Test for test results of omitting more images with small drones and birds

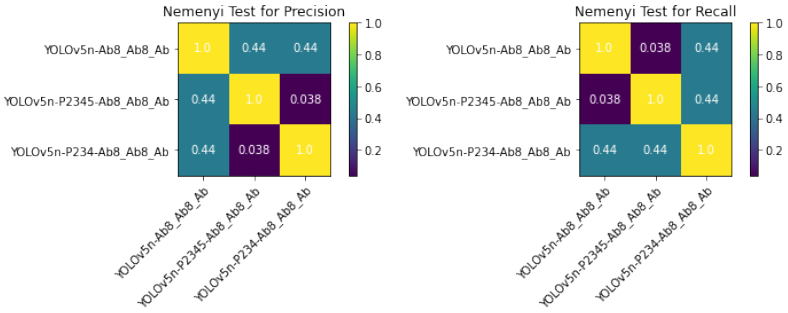
The performance of the model in detecting larger objects was better. The recalls even archived 1 when the threshold was larger than 16. However, as the precision increased, the recall of the complete test set became very low, which indicates that the model failed to detect small drones when trained without images of small drones and birds. In the Nemenyi test of precision, the precision of Ab32_Ab32_Ab was significantly higher than Ab8_Ab8_Ab, showing that the model was less likely to identify birds as drones for larger objects. By manually removing the small objects from the test set, the recall of Ab32_Ab32_Ab32 was significantly higher than Ab32_Ab32_Ab.

4.5 Impact of object size and additional detection head

In Section 4.1.2, the additional detection head did not work. In the above experiments, the YOLOv5-P2345 and YOLOv5-P234 models were revisited to see if they could perform better in recognizing drones and birds. In Table 4.14, YOLOv5n-P2345 had better precision when objects with an average sum of width and height smaller than 16 were removed, while the recall was still lower than YOLOv5n. In Nemenyi tests, the precision of YOLOv5n-P2345 was significantly higher than YOLOv5-P234 when trained with Ab8 and Ab16, which shows the importance of detection head for large objects. In Figure 4.22a,when more small objects were omitted in the train set, the precision of YOLOv5n was significantly higher than YOLOv5n-P2345, while there was no statistically significant difference between YOLOv5n-P2345 and YOLOv5n-P234.

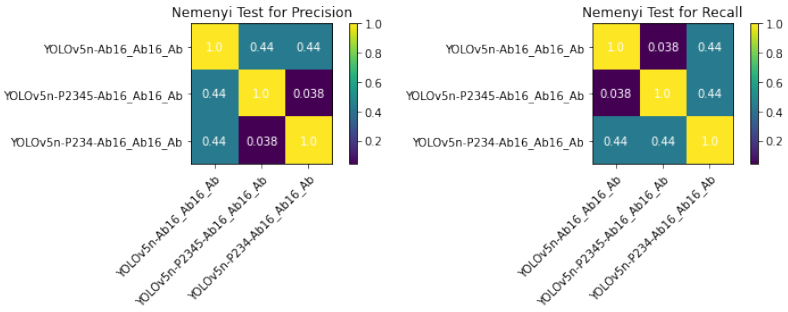
Table 4.14: Test results of YOLOv5n model and its variations

Model	Precision (%)	Recall (%)
YOLOv5n-Ab8_Ab8_Ab	78.5 ± 0.3	**71.6 ± 0.5**
YOLOv5n-P2345-Ab8_Ab8_Ab	**86.5 ± 0.2**	63.8 ± 0.3
YOLOv5n-P234-Ab8_Ab8_Ab	77.0 ± 0.3	68.8 ± 0.2
p-value of the Friedman test	0.049	0.049
YOLOv5n-Ab16_Ab16_Ab	84.8 ± 0.2	**64.0 ± 0.6**
YOLOv5n-P2345-Ab16_Ab16_Ab	**95.8 ± 0.3**	54.4 ± 0.1
YOLOv5n-P234-Ab16_Ab16_Ab	84.5 ± 0.2	61.9 ± 0.2
p-value of the Friedman test	0.049	0.049
YOLOv5n-Ab32_Ab32_Ab	**93.5 ± 0.2**	**56.6 ± 0.2**
YOLOv5n-P2345-Ab32_Ab32_Ab	92.9 ± 0.2	54.3 ± 0.4
YOLOv5n-P234-Ab32_Ab32_Ab	85.2 ± 0.2	53.6 ± 0.4
p-value of the Friedman test	0.049	0.049



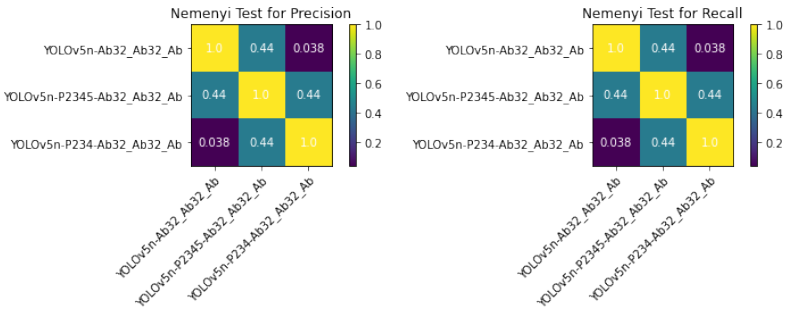
(a) (b)

Figure 4.20: Nemenyi Test for test results of YOLOv5n model and its variations (-8)



(a) (b)

Figure 4.21: Nemenyi Test for test results of YOLOv5n model and its variations (-16)



(a) (b)

Figure 4.22: Nemenyi Test for test results of YOLOv5n model and its variations (-32)

Chapter 5

Discussion

The experiments in this thesis were designed to combine the findings in related works in the drone detection area and to investigate more in detail. In [6], the author tested YOLOv3 and concluded that YOLOv3 yielded the overall best performance. The first experiment compares the models in the YOLO family. Besides YOLOv3 and YOLOv4, YOLOv5 of different sizes were included in the experiments, providing a full picture of the model performance in YOLO family, and bridging the gaps between the latest model versions and previous model benchmarking and research. The detection heads of the YOLOv5n model were then added or deleted, and the performance of the models was compared. The second experiment focused on the tiling method used by one of the teams the 2020 Drone vs. Bird Detection. YOLOv5n was used on the tiled dataset because it performed better than YOLOv3 in inference speed in the previous experiment. In the next experiment, the real dataset, a synthetic drone dataset and datasets taken by phones were used to compare the how the model performance would differ when trained with different dataset. The final experiment added annotated bird images into the dataset, and compared the model performance uder different training strategies. The results of the experiments will be discussed in the following sections.

5.1 SOTA model selection

5.1.1 YOLO family benchmarking

It is surprising to see that all YOLO models had good performance on detecting drones in AXIS dataset, especially when almost all the drones in the dataset were extremely small, which only took less than 0.003% area of the image.

That may be because the dataset used in this experiment only involved the drone class. Moreover, although the images was taken in sunny, cloudy and snowy weathers, most of the dataset was taken in good illumination condition. When comparing the models in YOLO family, we can find that YOLOv5 models generally outperform previous models in either precision, recall, or inference speed aspects. All YOLOv5 models have impressive speed. It only took the tiny model 7.9 ms to do detection on an image. The large model had roughly the same inference speed as YOLOv3, but its precision and recall increased by 0.057 and 0.013. YOLOv5s and YOLOv5m had a good balance in all the evaluation metrics. Although the inference speeds of YOLOv5s and YOLOv5m were slower than the speed limit for the tiling approach (12.5ms), they were fast enough to do inference on whole images in real-time.

5.1.2 Impact of adding or deleting detection heads

In Section 4.1.2, models with an additional detection head on shallow feature maps were tested. The inference time increased due to the extra regression time in the new detection head. After the detection head for the large object was deleted, the inference speed was still slower than the original inference speed. That was because the feature maps in shallow layers were bigger and took longer time to regress. However, the precision and recall decreased unexpectedly. In Section 4.5, the approaches of adding and deleting detection heads were revisited. It turned out that after removing objects whose average sum of width and height was smaller than 8 or 16, the detection head for extremely small objects helped improve the precision of the model. However, it would decrease recall. When there were only objects whose average sum of width and height was larger than 32, the small detection head began to lose its effect and even introduce more false positives. When tested with images containing all sizes of objects, the original YOLOv5n model had the most balanced performance. So using the default YOLOv5 structure is recommended when it may encounter objects of various sizes.

5.2 Impact of tiling

The YOLOv5n model trained in Section 4.1.1 was used for testing the tiled images. The relative object sizes in the train set were much smaller than the test set, so the features might differ a lot. As a result, the precision dropped from 0.944 to 0.736, which indicated that the features the model extracted were not representative of drones. When the model was trained and tested with tiled

images, the recall increased from 0.729 to 0.958. In the 2020 Drone vs. Bird Challenge, the AP reached 89.1 after tiling, but the comparison between the model trained before and after tiling was not shown [27]. Here the AP of YOLOv5n on AXIS dataset, though not shown in the result section, reached 97.5 from 78.2, which increased the performance by 24.6%, which was a huge improvement. Despite the little decrease in precision, training and testing with tiled images improved the performance of the model. The model was able to detect more drones of different backgrounds and sizes. The next model was trained without the pure background images in tiled datasets. Since there was only one drone in each original-size image, after tiling, an image would produce 2 to 7 tiles with pure backgrounds. In our tiled AXIS dataset, there were 4,274 tiles with a drone and 19,726 tiles of pure background. Compared with the model trained with the whole tiled AXIS dataset, the performance of the model trained only with 4,274 tiles was good actually. The precision only decreased by 0.012, and the recall even increased by 0.009. One thing we need to notice is that most of the time, the background of a drone was the sky in a tile. Omitting the other background images prevented the model to learn what features were not a drone, making the precision decrease. However, the good performance of the last model still shows the excessive background images were redundant, and it was enough for the model to achieve a good result without the background images.

5.3 Impact of camera

Due to the high resolution of the Phone4K images and the removal of pure background images, most of the tiled Phone4K images only contained a drone in the sky. Compared with the tiled AXIS and tiled PhoneHD datasets, the tiled Phone4K dataset lacked backgrounds such as forests and buildings. That made the precision of the model much lower. Moreover, the relative sizes of the drones in the tiled Phone4K dataset were larger, so the model was easier to miss small drones. The tiled PhoneHD dataset had more similar object sizes and contained comparatively sufficient backgrounds, so it had higher precision and recall. However, there was still a gap in having the same performance as the mode trained with the tiled AXIS dataset. The SimUAV was very different in relative object sizes. Because it was a simulated dataset for small drone detection, the drones were much smaller than our dataset, and very few large drones were included. Besides, the Mavic and the backgrounds in the images were also distinct from the real world. The images seemed to be over-sharpened and overexposed. Such differences between the train and

test sets made the model performance very poor. Contrary to the increment in model performance in [27], combining two dataset would decrease the model performance. That may be due to the test set and the train set came from the same sort, and the AAA model was over-fitted. The introduce of images taken by phones made the model less over-fitted to the AXIS images.

The experiment among different datasets show that training and testing on the same dataset had the best result, so if Skysense wants to implement the model on a different camera, it's better to retrain the model with the images taken from the new camera, especially when the resolutions of the cameras are different. The other experiment was designed to test if we could take advantage of the old dataset by adding images from new cameras. Assume that the old dataset is the tiled PhoneHD dataset. The new camera is AXIS. The performance of the model was tested when different amounts of tiled AXIS images were added to the train set. The results in Table 4.8 show that the more images from the new camera, the better the performance. The recall was higher when new images took up more than $\frac{4}{9}$ of the train set, while the precision still could not reach 0.923, the value when trained with a pure tiled AXIS dataset. So constructing a new dataset is still necessary when Skysense wants to replace AXIS with a new camera.

5.4 Impact of adding bird images and annotations

When bird images were added to the test set, the previous model detected many birds as drones, making precision drop a lot. The recall remained the same since there were no changes to the true positives or false negatives. When the bird images were also added to the train set, the decrease in both precision and recall showed that the model cannot learn the difference between bird and drone well. Most false positives and false negatives occurred when drones were extremely small. Some false positives came from the background buildings and trees. Results in Table 4.12 show, removing images with extremely small labels help improve the precision of the model. In [27], the birds were not annotated in the dataset, and here training drones with unannotated and annotated birds were both tested. Training birds as background had better results, which indicates that a lot of time on annotating the birds could be saved.

In Figure 4.17 and Figure 4.18, the average sums of the bird's and the drone's width and height were both 10 pixels. The confidence in predicting the

bird as a drone became lower when the small objects were removed from the train set, improving the precision. The confidence in predicting a true positive drone became lower as well, but the confidence remained high in Figure 4.18c. However, the small drone could not be detected in Figure 4.18d, so the recall dropped a lot. In Table 4.13, the precision and recall became very high when the sum of an object's average sum of width and height was larger than 16 pixels. That was useful because if Skysense had a very low tolerance for false positives caused by birds, the model (Tiled AXIS + birds as background (-16)) can be used to do the detection, and not confirm the detection until the sum of the bounding box's average sum of width and height is larger than 16 pixels.

Chapter 6

Conclusions and Future work

This chapter presents the conclusions and future work of this thesis.

6.1 Conclusions

The research questions proposed in Section 1.2 are as follows:

- What metrics are appropriate for evaluating drone detection performance?
- Which state-of-the-art model is best for real-time drone detection?
- How to improve the detection performance of small drones?
- How to reduce false positives caused by birds?

According to the requirements from Skysense, the model should be fast to run in real-time and be able to locate the drone location in the video stream. As a result, inference speed, precision and accuracy were selected as evaluation metrics. In the UAV detection benchmarking [6], YOLOv3 had the best inference speed and overall precision. Further benchmarking has been done in this thesis to test the performance of the latest YOLO models. The results of the benchmarking show that YOLOv5n and YOLOv5m are able to run in real-time with good performance, and YOLOv5n is the only model that is fast enough to run on image tiles in real-time. Tiling images can increase recall, avoiding the miss of small drones in detection. Adding an additional detection head can increase the precision, but it had a negative effect when trained with extremely small drones. To achieve the best performance, the train, validation, and test images should be taken by the same camera. Images taken from

other cameras can help improve recall but decrease the precision. Adding bird images as background in the train set can help improve the precision and recall, but extremely small drones and birds in the train set have a negative impact on model performance. The model is able to differentiate drones and birds better when the average sum of birds' or drones' width and height is larger.

6.2 Future work

Due to the limitation of available drones, YOLOv5 with different training strategies was tested on drone datasets that only contained DJI Mavics. To verify the generality of the conclusions of the thesis, experiments on datasets with drones of different types should be done in the future. Moreover, although the images was taken in sunny, cloudy and snowy weathers, most of the dataset was taken in good illumination condition. Images taken in rainy days or at dawn and dusk are expected to be added into the dataset, too.

This thesis focuses on improving the model performance on images taken by one specific camera that was used in Skysense, so the images in the test set were all taken by the AXIS camera. Further study could try constructing the test set from different sources and investigating the generalization capabilities of the model and the corresponding improvement.

References

[1] "Sweden drones: Sightings reported over nuclear plants and palace," *BBC*. [Online]. Available: https://www.bbc.com/news/world-europe-60035446 [Page 1.]

[2] M. Z. Anwar, Z. Kaleem, and A. Jamalipour, "Machine learning inspired sound-based amateur drone detection for public safety applications," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 3, pp. 2526–2534, 2019. doi: 10.1109/TVT.2019.2893615 [Page 5.]

[3] G. J. Mendis, T. Randeny, J. Wei, and A. Madanayake, "Deep learning based doppler radar for micro uas detection and classification," in *MILCOM 2016 - 2016 IEEE Military Communications Conference*, 2016. doi: 10.1109/MILCOM.2016.7795448 pp. 924–929. [Page 5.]

[4] S. R. Ganti and Y. Kim, "Implementation of detection and tracking mechanism for small uas," in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2016. doi: 10.1109/ICUAS.2016.7502513 pp. 1254–1260. [Page 5.]

[5] E. Unlu, E. Zenou, and N. Rivière, "Generic fourier descriptors for autonomous uav detection," in *ICPRAM*, 2018. [Pages 5 and 6.]

[6] B. K. S. Isaac-Medina, M. Poyser, D. Organisciak, C. G. Willcocks, T. P. Breckon, and H. P. H. Shum, "Unmanned aerial vehicle visual detection and tracking using deep neural networks: A performance benchmark," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, October 2021, pp. 1223–1232. [Pages 5, 6, 7, 11, 47, and 53.]

[7] E. Unlu, E. Zenou, N. Rivière, and P.-E. Dupouy, "Deep learning-based strategies for the detection and tracking of drones using several cameras," *IPSJ Transactions on Computer Vision and Applications*,

vol. 11, no. 1, Dec. 2019. doi: 10.1186/s41074-019-0059-x. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02863410 [Page 5.]

[8] S. R. Ganti and Y. Kim, "Implementation of detection and tracking mechanism for small uas," in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2016. doi: 10.1109/ICUAS.2016.7502513 pp. 1254–1260. [Page 5.]

[9] A. Shoufan, H. M. Al-Angari, M. F. A. Sheikh, and E. Damiani, "Drone pilot identification by classifying radio-control signals," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 10, pp. 2439–2447, 2018. doi: 10.1109/TIFS.2018.2819126 [Page 5.]

[10] Y. Liu, L. Liao, H. Wu, J. Qin, L. He, G. Yang, H. Zhang, and J. Zhang, "Trajectory and image-based detection and identification of uav," *Vis. Comput.*, vol. 37, no. 7, p. 1769–1780, jul 2021. doi: 10.1007/s00371-020-01937-y. [Online]. Available: https://doi.org/10.1007/s00371-020-01937-y [Page 6.]

[11] Z. Wang, L. Qi, Y. Tie, Y. Ding, and Y. Bai, "Drone detection based on fd-hog descriptor," in *2018 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, 2018. doi: 10.1109/CyberC.2018.00084 pp. 433–4333. [Page 6.]

[12] A. Canepa, E. Ragusa, R. Zunino, and P. Gastaldo, "T-rexnet—a hardware-aware neural network for real-time detection of small moving objects," *Sensors*, vol. 21, no. 4, 2021. doi: 10.3390/s21041252. [Online]. Available: https://www.mdpi.com/1424-8220/21/4/1252 [Page 6.]

[13] A. Tanzia, R. Tanvir, B. R. Bir, and U. Jia, "Drone detection by neural network using glcm and surf features," in *2021 Journal of Information Systems and Telecommunication (JIST)*, vol. 0, 2021, pp. 15–24. [Page 6.]

[14] R. Jiang, Y. Zhou, and Y. Peng, "A review on intrusion drone target detection based on deep learning," in *2021 IEEE 4th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, vol. 4, 2021. doi: 10.1109/IMCEC51613.2021.9482092 pp. 1032–1039. [Pages 6 and 10.]

[15] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014. [Page 7.]

[16] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015. [Page 7.]

[17] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28. Curran Associates, Inc., 2015. [Online]. Available: https://proceedings.neurips.cc/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf [Page 7.]

[18] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *CoRR*, vol. abs/1804.02767, 2018. [Online]. Available: http://arxiv.org/abs/1804.02767 [Pages 7, 8, 10, and 19.]

[19] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016. ISBN 978-3-319-46448-0 pp. 21–37. [Page 7.]

[20] X. Chen, J. Lv, Y. Fang, and S. Du, "Online detection of surface defects based on improved yolov3." in *Sensors (Basel)*, 2022. doi: 10.3390/s22030817 [Page 8.]

[21] A. Bochkovskiy, C. Wang, and H. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *CoRR*, vol. abs/2004.10934, 2020. [Online]. Available: https://arxiv.org/abs/2004.10934 [Page 9.]

[22] G. J. et. al., "ultralytics/yolov5: v6.0 - YOLOv5n 'Nano' models, Roboflow integration, TensorFlow export, OpenCV DNN support," Oct. 2021. [Online]. Available: https://doi.org/10.5281/zenodo.5563715 [Page 9.]

[23] C. Rui, G. Youwei, Z. Huafei, and J. Hongyu, "A comprehensive approach for uav small object detection with simulation-based transfer learning and adaptive fusion," 2021. [Pages 10 and 12.]

[24] F. . Ünel, B. O. Özkalayci, and C. Çiğla, "The power of tiling for small object detection," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019. doi: 10.1109/CVPRW.2019.00084 pp. 582–591. [Page 10.]

[25] E. Unlu, E. Zenou, N. Riviere, and P.-E. Dupouy, "Deep learning-based strategies for the detection and tracking of drones using several cameras," *IPSJ Transactions on Computer Vision and Applications*, vol. 11, 12 2019. doi: 10.1186/s41074-019-0059-x [Page 11.]

[26] Y. Lv, Z. Ai, M. Chen, X. Gong, Y. Wang, and Z. Lu, "High-resolution drone detection based on background difference and sag-yolov5s," *Sensors*, vol. 22, no. 15, 2022. doi: 10.3390/s22155825. [Online]. Available: https://www.mdpi.com/1424-8220/22/15/5825 [Page 11.]

[27] A. Coluccia, A. Fascista, A. Schumann, L. Sommer, A. Dimou, D. Zarpalas, M. Méndez, D. de la Iglesia, I. González, J.-P. Mercier, G. Gagné, A. Mitra, and S. Rajashekar, "Drone vs. bird detection: Deep learning algorithms and results from a grand challenge," *Sensors*, vol. 21, no. 8, 2021. doi: 10.3390/s21082824. [Online]. Available: https://www.mdpi.com/1424-8220/21/8/2824 [Pages 11, 49, and 50.]

[28] R. Yoshihashi, R. Kawakami, M. Iida, and T. Naemura, "Bird detection and species classification with time-lapse images around a wind farm: Dataset construction and evaluation," *Wind Energy*, vol. 20, pp. 1983–1995, 2017. [Pages 12 and 13.]

[29] S. Herbold, "Autorank: A python package for automated ranking of classifiers," *Journal of Open Source Software*, vol. 5, no. 48, p. 2173, 2020. doi: 10.21105/joss.02173. [Online]. Available: https://doi.org/10.21105/joss.02173 [Page 18.]

TRITA-EECS-EX- 2022:32