Degree Project in the Field of Technology Electrical Engineering and the Main Field of Study ICT Innovation

Second cycle, 30 credits

Investigating Altitude-Adaptive Methods for Enhancing Small Object Detection on UAVs

PHILIPP AHRENDT

Investigating Altitude-Adaptive Methods for Enhancing Small Object Detection on UAVs

PHILIPP AHRENDT

Master's Programme, ICT Innovation, 120 credits
Date: October 17, 2024

Supervisors: Jörg Conradt, Marco Moletta
Examiner: Arvind Kumar
School of Electrical Engineering and Computer Science
Host company: FLOX Robotics
Swedish title: Undersökning av höjdadaptiva metoder för förbättrad detektering av små objekt på UAVs

Abstract

Computer vision is vital for the recent development of aerial vision-based applications, however, small object detection remains a challenge even for state-of-the-art models such as YOLOv7-tiny. Object detection from Unmanned Aerial Vehicles (UAVs) is particularly affected because of their sometimes high flight altitude leading to a greater amount of small objects to detect, while their limited storage and computing power capacity restricts the model complexity. Contextual information such as the flight altitude is often readily available from onboard sensors and can serve as relevant prior knowledge to a neural network learning to predict the size, location, and class of objects in an image. Despite that, research in this direction is sparse and focuses on specific applications, demanding more general approaches. This thesis investigates various methods to integrate altitude information into the learning process of an object detection network. The focus of this work lies in analyzing the influence of the proposed methods on precision, recall, and mean average precision (mAP) for aerial datasets across different altitude levels and in general. We demonstrate that concatenating the input image with the altitude information or adding an auxiliary head that predicts the altitude from an image can help to slightly boost the performance. Furthermore, despite a reduced average precision, a dynamic loss based on altitude can offer more controlled fine-tuning of the model depending on the specific requirements of a UAV-based detection task. However, our results also showed that the greatest benefits stemmed from the addition of a small object detection head and the removal of the largest head which is unrelated to flight altitude. Nevertheless, this approach could potentially be further improved through one of the altitude adaptive methods since they are not mutually exclusive. The proposed methods and benchmarks provide a foundation for future research in the area of altitude-aware models as well as validate research on UAV-optimized YOLO models. Overall, this work provides an overview of how contextual information could be integrated into an existing object detection model and its effects on the training process and inference performance.

Keywords

Small Object Detection, Altitude adaptive, Unmanned Aerial Vehicles (UAV), YOLOv7-tiny

Sammanfattning

Datorseende är avgörande för den senaste utvecklingen av applikationer baserade på flygseende, men detektering av små objekt är fortfarande en utmaning även för toppmoderna modeller som YOLOv7-tiny. Objektdetektering från obemannade flygfarkoster (UAVs) påverkas särskilt på grund av deras ibland höga flyghöjd, vilket leder till en större mängd små objekt att upptäcka, samtidigt som deras begränsade lagrings- och datorkapacitet begränsar modellens komplexitet. Kontextuell information som flyghöjd är ofta lättillgänglig från sensorer ombord och kan fungera som relevant förkunskap för ett neuralt nätverk som lär sig att förutsäga storlek, plats och klass för objekt i en bild. Trots detta är forskningen i denna riktning sparsam och fokuserar på specifika applikationer, vilket kräver mer generella tillvägagångssätt. I den här avhandlingen undersöks olika metoder för att integrera höjdinformation i inlärningsprocessen för ett nätverk för objektdetektering. Fokus för detta arbete ligger i att analysera de föreslagna metodernas påverkan på precision, återkallande och genomsnittlig genomsnittlig precision (mAP) för flygdataset på olika höjdnivåer och i allmänhet. Vi visar att om man sammankopplar indatabilden med höjdinformationen eller lägger till ett extra huvud som förutsäger höjden från en bild kan det bidra till att öka prestandan något. Trots en minskad genomsnittlig precision kan en dynamisk förlust baserad på höjd dessutom erbjuda en mer kontrollerad finjustering av modellen beroende på de specifika kraven för en UAV-baserad detekteringsuppgift. Våra resultat visade dock också att de största fördelarna härrörde från tillägget av ett detekteringshuvud för små objekt och borttagandet av det största huvudet som inte är relaterat till flyghöjden. Trots detta kan denna metod potentiellt förbättras ytterligare genom någon av de höjdadaptiva metoderna eftersom de inte utesluter varandra. De föreslagna metoderna och riktmärkena utgör en grund för framtida forskning inom området höjdmedvetna modeller samt validerar forskning om UAV-optimerade YOLO-modeller. Sammantaget ger detta arbete en översikt över hur kontextuell information kan integreras i en befintlig objektdetekteringsmodell och dess effekter på träningsprocessen och inferensprestanda.

Nyckelord

Detektering av små objekt, Höjdadaptiv, Obemannade Flygfarkoster (UAV), YOLOv7-tiny

Acknowledgments

I would like to express my gratitude to Associate Professor Jörg Conradt for supervising my degree project and providing me with valuable feedback throughout this period. I am equally grateful to Marco Moletta and Kishore Kumar for closely following my progress, offering regular feedback, and supporting me through all stages of this thesis. I would also like to extend my thanks to Associate Professor Arvind Kumar for serving as my KTH examiner.

I am also grateful to FLOX Robotics and its team for giving me the opportunity and resources to work on this project.

Finally, I would like to thank my family and friends for their support and encouragment throughout this journey.

Stockholm, October 2024
Philipp Ahrendt

Contents

List of Figures

List of Tables

List of acronyms and abbreviations

AP	Average Precision
BCE	binary cross-entropy
CNN	Convolutional Neural Network
FLOPS	Floating Point Operations
FPN	Feature Pyramid Network
GPS	Global Positioning System
IoU	Intersection over Unit
mAP	Mean Average Precision
POG	PeopleOnGrass
UAV	Unmanned Aerial Vehicle

Chapter 1

Introduction

Computer vision is a branch of Artificial Intelligence that focuses on enabling computers to interpret visual data. Through the advancements of machine learning and deep learning and thanks to the development of better hardware, especially over the recent years, the field of computer vision has seen major improvements. While in the past, methods have relied heavily on manual feature kernels, modern computer vision tasks make use of data-driven approaches and neural networks.

While computer vision is a broad field, the recent research focus has been mainly on image classification or object detection through benchmarking on datasets like ImageNet [1] and COCO [2]. Object detection deals with the recognition and classification of items or entities such as humans, vehicles, obstacles, or animals in an image scene. Object detection is crucial and is usually one of the first steps for autonomous systems such as self-driving cars or Unmanned Aerial Vehicles (UAVs).

One of the challenges that has persisted over the years has been the detection of small objects [3]. Small objects are difficult to detect because they occupy a very small area and sometimes only as little as a few pixels in an image. To state some numbers COCO, a widely used dataset for object detection, defines small objects as 32x32 pixels or less in a commonly used image size of 480x640 [4], while in certain cases objects can also be as small as 5x5 pixels. Furthermore, state-of-the-art deep learning models use several convolutions to extract features from a scene, which causes spatial information to get lost in the process. All of this results in small objects being often missed or falsely localized during the detection process. This is especially relevant for object detection using UAVs, as taking pictures high up in the air causes objects to be small compared to the rest of the background. Furthermore, UAV-

based object detection faces even more challenges than just small objects, such as differences in viewpoints, the need for embedded algorithms, and real-time methods which put additional constraints on the type of models that can be designed and developed. Nevertheless, this is a relevant field that has its application from farming to search and rescue operations to environmental protection to much more.

However, the nature of UAVs also introduces components that can be beneficial if we analyze the problem holistically and if we integrate the context they provide. Intuitively, as humans, we make use of additional information and the environment we are in. As an example, looking out of an airplane we know that a house will look significantly smaller than when looking at one from the ground. As UAVs are not only cameras flying in the air but rather complex systems that possess a variety of sensors such as Global Positioning Systems (GPSs), barometers, and Inertial Measurement Units, they can have a similar or even better sense of context and environment than what we humans have.

1.1 Contributions

This thesis presents a comprehensive evaluation of UAV-based computer vision models for the detection of small objects, incorporating altitude data. It benchmarks the performance of the state-of-the-art model, YOLOv7-tiny, across multiple datasets and altitude levels and analyzes the results of the proposed methods according to this benchmark. Furthermore, this work explores and gives insights into how altitude information could be integrated into current object detection models as a dynamic training or learning-based method. While contextual information such as altitude holds informative value and can be applied through preprocessing techniques or expert models, the complexity increases when trying to integrate it in learning-based approaches. With this study we contribute to the understanding of multi-modal approaches in the space of object detection and what effects it can have on learning.

1.2 Problem

The problem this thesis addresses is related to the difficulty of detecting small objects when a UAV is flying at high altitudes. While small object detection is an intrinsic problem of computer vision, we will try to understand the factors in current state-of-the-art detectors that make this problem challenging and

further aim to understand the relationship between object size, altitude, and detection performance. In this work detection performance is defined through the average precision metric discussed in more detail in Section 3.7. Thus, the research question this work addresses is summarized as follows:

Is it possible to make use of altitude information to improve detection performance through dynamic or learning-based approaches?

1.3 Purpose

The main purpose of this work is to explore if the contextual information of altitude can be used in a dynamic or learning-based approach to improve object detection in UAVs with a generic method that works across a variety of aerial datasets. This results in evaluating a current state-of-the-art model as well as implementing possible adaptations to this model with the aim of improving specific metrics without decreasing other metrics or at least finding an acceptable balance.

1.4 Goals

The goals of this thesis can be summarized as follows:

- Implement and test various small object detection methods that make use of altitude information
- Analyze the findings of the experiments and propose different models that can be used in generic UAV-based object detection settings
- Test and compare the adapted models with their baseline and evaluate their performance

1.5 Research Methodology

The main research methods employed throughout this thesis are deductive reasoning and quantitative analysis of experiment findings. Quantitative data analysis helped prove the intuition that object size is correlated with altitude for aerial images. With this correlation observed, deductive reasoning was needed to work out methods that can incorporate this altitude information to help the model learn to better detect objects at varying altitudes. The analysis

of the results of the experiments was once again of quantitative nature as the evaluation methods were based on numeric values.

1.6 Ethics and Sustainability

From an ethical point of view, this research field and direction can be applied to wildlife detection as well as human detection by UAVs for search and rescue missions. Researching small object detection at higher altitudes could help improve the operational altitude level at which drones can still detect humans, increasing the area that can be covered per flight and thus possibly increasing the chances of finding the missing people.

Lastly, while current trends suggest that more data and more computing power can usually improve performance, from a sustainability aspect it is still relevant to develop solutions that are efficient by for example exploiting contextual knowledge instead of using more computing power to solve the problem. Improving models or allowing them to learn faster by encouraging the use of readily available metadata instead of increasing the model size or training them for longer periods for the same improvements can help save energy and incite more research in that direction.

1.7 Delimitations

The main delimitations of this work are of time and resource-based nature. With a limited number of research findings that integrate altitude in a dynamic or learning-based manner, this meant trying out various approaches with limited possible insights prior to running the experiments. Due to the nature of deep learning, the available data is another possible limitation concerning the quality and quantity of the data. Furthermore, while experiments could be run on a cloud server provided by FLOX Robotics, even allowing the number of experiments run in this work in the first place, there was still a limit to the amount of computing power and resources available to run experiments in parallel or for an extended amount of times.

1.8 Structure of the thesis

This work is structured as follows:

- **Chapter 2:** This chapter presents the literature review, which will cover an overview and analysis of UAV-based computer vision and a summary of modern object detection methods based on deep learning. It will also cover YOLOv7-tiny, the state-of-the-art algorithm used in this thesis and related works that are trying to solve small object detection in UAVs
- **Chapter 3:** Here, we explain the methods employed in this work and the reasoning behind the choices taken for the implementation and experiments
- **Chapter 4:** In this chapter we will analyze the data used, give a brief description of the experimental setup, and present the results.
- **Chapter 5:** Following the results we will discuss what they imply and what we were able to learn through this work.
- **Chapter 6:** Finally, we will conclude this work by summarizing the findings, raising some final considerations, and suggesting potential future works

Chapter 2

Background

This section covers the main background area necessary to understand this work which is object detection in UAVs. It also describes the works that are related to the methods carried out in this thesis.

More specifically, Section 2.1 covers the topic of object detection in UAVs and its challenges, while Section 2.2 describes the main three object detector categories commonly found in recent years. Section 2.3 explains how YOLOv7-tiny, a specific object detection model, works in more detail. Finally, Section 2.4 summarizes works that make use of dynamic anchors, optimize YOLO specifically for UAVs, or that use altitude-adaptive methods to improve object detection.

2.1 Object detection in UAVs

The field of object detection has seen major improvements over the last years which enabled the adoption of it in emerging fields such as the area of UAV. There are numerous applications where the use of computer vision or in particular object detection in UAVs can be of great benefit. This includes but is not limited to wildlife monitoring, navigation, surveillance, search and rescue operations [5]. To perceive their environment these drones can use different sensors such as cameras or lidars and perform their object detection tasks either through embedded devices on-board or by sending the data to a ground station for processing. On top of including the images for object detection, this data could also include additional meta-data of other onboard sensors of the drone. Some examples of this would be altitude data from a barometer, location data from a GPS, or camera angle from a gyrometer.

2.1.1 Altitude

In the context of this work, altitude refers to the vertical distance of a UAV to the ground level. Altitude can be measured by instruments such as barometers, altimeters, or GPS systems [6]. While in general barometers only measure the air pressure in a given environment, the most common types of altimeters are barometric, meaning they use that information to determine the altitude of an object. GPS systems can also be used to indicate the altitude relative to the sea level through triangulation of the sensor with satellites. Both of the sensors have their advantages and challenges. While altimeters do not require a clear view of the sky like GPS systems do, they can be influenced by air temperature and humidity for example [7]. GPS systems on the other hand can be affected by signal interference and satellite availability [8], while also being heavier and more expensive in general. Most UAV systems have either one or both of the sensors integrated and this information is captured during flight time and readily available for diagnosis or to be used in related applications. For the use case of object detection, this altitude information can be an important prior knowledge indicator about the size of the objects.

For UAV applications there can be a significant variability in altitude. For example in traffic surveillance the flight altitude for the UAV is said to be in the range of 5 to 100 meters [9], while for search and rescue operations the altitude can go up to 260 meters [10]. This leads to a significant variance in size for the same class of objects and thus requirements and challenges for a good UAV-based object detector.

2.1.2 Challenges

On top of the challenge of small objects which make it hard to distinguish them from the background, UAVs also have limited computing power and storage space since they are small and lightweight [11]. This makes it more difficult to run complex algorithms in real time on them. Furthermore, UAVs also face the challenge of significant variance in lighting and weather conditions which affects the accuracy of the algorithms [12]. Moving objects is another difficulty, as tracking moving objects in a dynamic environment such as surveillance is more challenging than in static conditions. While some of these challenges are not only inherent to UAVs, their limited capacity and the conditions they operate under — such as higher altitudes, outdoors, and a broader range of scenarios — make it increasingly important to address these issues.

2.2 Evolution of Object Detection Models

Object detection deals with the task of recognizing and localizing all objects in an image. This is done by training an object detector with a labeled dataset of images and corresponding annotations, The annotations specify the classes of all object instances and describe their location through a bounding box that encapsulates them The object detection model then learns how to predict the predefined classes and the bounding box through a generally supervised learning method.

When going through the literature of recent years, object detectors can be in general categorized into one of two methods: Two-stage detectors and one-stage detectors. While methods are usually based on convolutional neural networks, more recently a new class of object detectors emerged that is transformer-based.

2.2.1 Two-stage detectors

As the name implies two-stage object detectors work through a two-step process involving region proposal generation and object detection. The first stage makes use of region proposal networks to identify potential object locations in the image by generating region proposals. The proposed candidate regions are then refined and classified in the second stage to determine the presence of objects and their corresponding classes. Two-stage detectors often leverage features from deep Convolutional Neural Network (CNN) backbones to extract and process image features. Examples of these detectors include Feature Pyramid Network (FPN) [13] and Faster-RCNN [14]. While two-stage detectors tend to have higher accuracy, their structure makes them larger in size and slower in processing and thus less suitable for real-time applications.

2.2.2 One-stage detectors

In contrast to the two-stage detectors, one-stage detectors perform object detection in a single step without the need for region proposals. They directly predict the location of the bounding boxes and class probabilities for the objects in an image. These detectors typically use a single CNN architecture to process the entire image at once and generate detection outputs. Examples of these types of object detectors include YOLO [15] and SSD [16] and while they might sacrifice some detection accuracy especially on smaller or greatly occluded objects, they are generally faster and more lightweight. This is the

reason why they are more suitable for onboard scenarios.

2.2.3 Transformer-based detectors

In more recent years, the transformer architecture has risen in popularity in a variety of applications [4]. While a type of architecture originally developed for natural language processing tasks, they have been adapted for computer vision tasks, including object detection, by incorporating the self-attention mechanism. The self-attention mechanism allows the model to focus on different parts of the input sequence and can help the transformer to capture long-range dependencies and relationships between objects in an image which has the potential to improve object detection performance. Transformers typically rely on the encoder-decoder architecture where, in object detection, the encoder extracts features from the input image and the decoder produces the detection output.

While vision transformer models such as [17] show promise in the field of object detection their architecture is commonly even larger in size and slower in inference speed than two-stage detectors making them unsuitable for UAV-based real-time detection applications so far.

2.3 YOLOv7-tiny

YOLOv7 was first published by Wang *et al.* [18] to surpass all known object detectors in speed and accuracy in the real-time object detector landscape. The authors proposed multiple methods and architecture changes to improve the performance of the previous YOLO iterations. Furthermore, they developed multiple versions of the same YOLOv7 architecture depending on the width and depth of the network with YOLOv7-tiny being the smallest and fastest one. While the larger YOLOv7 or YOLOv8 models offer improvements in accuracy, they come with increased computational demands and inference times. As some embedded devices, but in particular UAVs are dependant on the speed of the object detection the choice for a model that has high accuracy yet is real-time and a memory footprint that allows it to run on these devices YOLOv7-tiny has become one of the most mainstream object detection algorithms currently. The fact that this model is one of the state-of-the-art algorithms in aerial object detection can be seen by the number of papers that try to improve on it which are described in more detail in Section 2.4. While some two-stage detectors promise better accuracy they are not fast enough for real-time applications and other one-stage detectors are too big

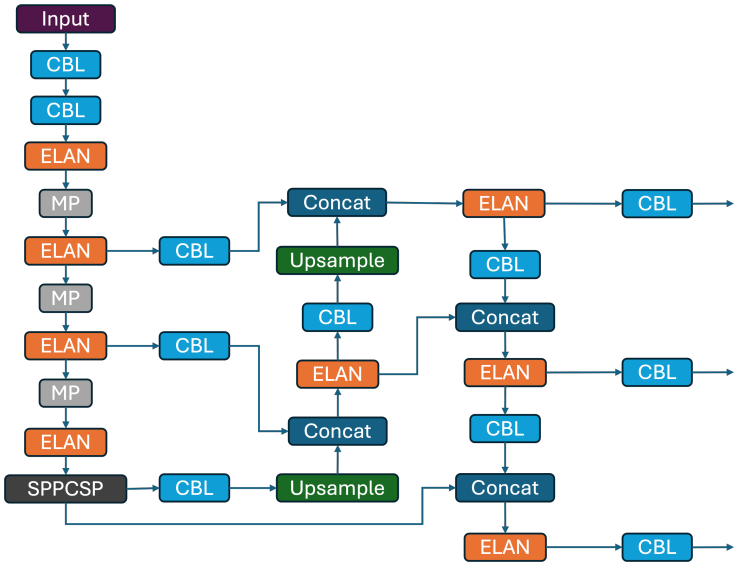


Figure 2.1: YOLOv7-tiny architecture

for deployment on a drone. Furthermore, as there is ample research that uses YOLOv7-tiny as a baseline, it provides some additional comparability.

The architecture of YOLOv7-tiny can be seen in Figure 2.1. It follows the layout of previous iterations with a backbone to extract features from an image through convolutions. The next part is the neck or feature fusion layer which links the extracted features from different layers of the backbone and helps pass semantic information from deeper layers to the more spatially-aware shallower layers which capture more spatial information and the reverse as well. In YOLOv7-tiny the neck architecture consists of an FPN. At the end of the network are three detection heads that work on feature maps of different scales. This results in the first and most shallow head being better at detecting small objects, the middle head being better for medium-sized objects and the deepest head being best at detecting the largest objects.

2.3.1 IoU

Intersection over Union is a metric that quantifies how much overlap exists between two bounding boxes. In the case of object detection model that is the overlap between a predicted and the ground-truth bounding box. It is defined

as the area of intersection of the two boxes divided by their area of union:

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}} \tag{2.1}$$

The IoU is needed to calculate the loss of YOLOv7 and also important to evaluate the models performance as detailed in Section 3.7.

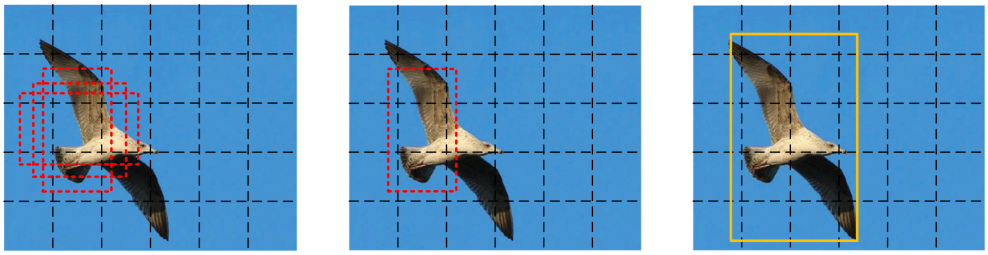
2.3.2 Anchors

One common method that many object detectors such as the YOLO models or other object detection systems such as SSD [16] or Faster-RCNN [14] use are anchor boxes. Sometimes referred to as grid cells [15], they are like predefined bounding boxes specified through their size and aspect ratio that help the network regress a set of bounding boxes along with the class. This works by assigning the anchors to objects and learning the offset to the ground-truth bounding boxes [14].

Anchor boxes were integrated into the YOLO models starting from the second version with each iteration improving on the mechanism. YOLOv7 uses 9 anchor boxes, with 3 boxes for each prediction head. Each predefined anchor box has a different size and aspect ratio to fit to various object scales in a dataset. The image is divided into grids that corresponding to the dimensions of the feature map of the last layer at each prediction head. Each prediction head is responsible for detecting objects at different scales, and each grid cell within these heads is assigned a set of anchor boxes. The heads predict the offset to these anchors to match the position and size of the objects in the image. During training the ground truth boxes will be matched to the anchors with the highest Intersection over Unit (IoU) so that the model can learn to fit the anchors closely to the ground truth boxes. IoU is a metric that measures the overlap between two bounding boxes. For each grid cell the model will predict whether or not there is an object (objectness score), the offset to the anchors and the class of the object. An illustration of this process can be seen in Figure 2.2.

2.3.3 Loss function

The total loss function used in YOLOv7-tiny can be separated into three components: the bounding box loss, the objectness loss, and the classification loss. Each of these components play a role in the accurate detection, localization and classification of objects in an image.



(a) (b) (c)

Figure 2.2: (a) Default anchor boxes (red) are placed on each grid cell of the feature map and projected onto the input image. (b) A subset of anchor boxes is selected based on the score predicted by the network. (c) The selected anchor box is adjusted according to the network's predicted offsets resulting in the final predicted box. Reprinted from "Adaptive anchor box mechanism to improve the accuracy in the object detection system" [19].

The total loss function can be summarized as follows:

$$\mathcal{L}_{total} = \gamma_{box} \cdot \mathcal{L}_{box} + \gamma_{obj} \cdot \mathcal{L}_{obj} + \gamma_{cls} \cdot \mathcal{L}_{cls} \tag{2.2}$$

where γ_{box}, γ_{obj}, and γ_{cls} are tunable hyperparameters that control the importance of each loss component.

2.3.3.1 Box Loss

The bounding box loss measures the localization error between the predicted bounding box and the ground truth bounding box. This error is based on the IoU and more specifically CIoU [20] for YOLOv7, taking into account the overlap, aspect ratios and distance between the center of the boxes. It's defined as:

$$\mathcal{L}_{box} = 1 - \text{CIoU}(b_p, b_{gt}), \tag{2.3}$$

where b_p is the predicted bounding box, and b_{gt} is the ground truth bounding box. The CIoU loss is defined as:

$$\text{CIoU}(b_p, b_{gt}) = \text{IoU}(b_p, b_{gt}) - \frac{\rho^2(b_p, b_{gt})}{c^2} - \alpha \cdot v, \tag{2.4}$$

where ρ represents the Euclidean distance between the central points of the bounding boxes, c denotes the diagonal length of the smallest box that encompasses the two boxes, α is a positive trade-off parameter and v measures the consistency of the aspect ratio.

2.3.3.2 Objectness Loss

The objectness loss term measures how confident the network is about an object existing inside a bounding box prior. It is calculated using binary cross-entropy (BCE) between the predicted objectness score and the ground truth:

$$\mathcal{L}_{obj} = -\sum_{i=1}^{N} \left[t_i^{obj} \log(p_i^{obj}) + (1 - t_i^{obj}) \log(1 - p_i^{obj}) \right], \tag{2.5}$$

where N is the number of anchors, t_i is the ground truth (1 if an object exists, 0 otherwise) and p_i is the predicted objectness score for anchor i.

2.3.3.3 Classification Loss

Finally, the classification loss measures the error in the class prediction. It is also calculated through BCE:

$$\mathcal{L}_{cls} = -\sum_{i=1}^{N} \sum_{c=1}^{C} \left[t_{i,c}^{cls} \log(p_{i,c}^{cls}) + (1 - t_{i,c}^{cls}) \log(1 - p_{i,c}^{cls}) \right], \tag{2.6}$$

where C specifies the number of classes, $t_{i,c}$ the class label, and $p_{i,c}$ the predicted class probability for anchor i and class c.

2.4 Related Work

Object detection in aerial images can be quite different compared to object detection in natural images which are typically captured by cameras on the ground. As seen in Figure 2.3 objects such as humans occupy a significantly smaller area which translates to a smaller number of pixels and can thus be more challenging for object detectors to identify. This Section explores three different types of work that are related to the topic of this thesis.

(a) (b)

Figure 2.3: Comparison of a natural image (a) from the COCO dataset [2] and an aerial image (b) taken by a drone high up in the air. We can see that the view angle is different and that the people in the center we wish to detect take only a small part of the image. Furthermore, at this angle there is less of a distinction between foreground and background, the objects are part of a complex scene.

2.4.1 Dynamic Anchors

Commonly, anchor boxes are either meticulously hand-crafted or defined after running a clustering algorithm. Some methods in the past have been proposed to improve upon these pre-defined anchors. Meta-anchor is a method that dynamically generates anchor functions from arbitrary customized prior boxes to improve the robustness of anchor-based object detector to anchor settings and bounding box distribution [21]. Zhang *et al.,* developed Free-anchor, which updates hand-crafted anchors, but this method models detector training as a maximum likelihood estimation to optimize the matching of objects and anchors [22].

”Adaptive Anchor for Fast Object Detection in Aerial Image” is one such method, where the authors aimed to improve the detection performance by proposing a scale-aware network. This network learns to determine the scale of predefined anchors thereby reducing the scale search range and the risk of overfitting as well as improving detection accuracy and speed [23]. While the authors do not make use of the image capture height explicitly, they mention its relevance to motivate their research.

While these methods are slightly older and with the improvement of the network architecture and the addition of methods such as employing an evolutionary algorithm to find good anchors in YOLOv7, it is unclear whether the proposed methods still hold any advantage for current state-of-the-art object detectors. Nevertheless, they provide some evidence that in some cases a dynamic approach to pre-defined anchors might be of benefit, and with more

recent works on anchor configuration improvement such as the IoU-aware clustering [24], there is still room for improvement. An approach using some of these insights was investigated for dynamic anchors in an altitude-adaptive approach in Section 3.5.1.

2.4.2 Optimized UAV YOLO models

To address the challenges described in Section 2.1 a multitude of papers have recently been published proposing methods to improve YOLOv7 and related models specifically for the use case of UAVs. A common denominator these works worked out was integrating an additional head at a shallower layer and even removing the head of the deepest layer [25]. This falls in line with the argument that shallower heads are better at detecting smaller objects which is one of the main challenges in object detection in UAVs.

Another change that has been proposed multiple times was the adoption of alternate loss functions. Wang *et al.,* applied the SIoU loss function for better performance [26], while Hu et al introduced α-CIoU to address sample imbalance [11]. Yet another loss function, the Wise-IoU [27], designed to improve the bounding box regression by taking into account the quality of the anchor boxes adopted in the SMFF-YOLO [28] and the PDWT-YOLO [29].

A third change proposed in some of these papers was an adaptation of the feature fusion layer of the YOLO architecture also known as the neck of the network. The reason as to why these changes were introduced ranged from handling scale variations and complex backgrounds [28] to enhancing feature fusion capabilities [26][30] and augmenting the representation of small object features [11].

While the referenced papers aim to improve small object detection in UAVs, they primarily do so by introducing new components, specifically designed to address challenges like small objects or sample quality, into particular model architectures. An altitude-adaptive approach, however, could potentially be integrated into a variety of architectures and combined with these modifications for further enhancement.

A summary of the described papers can be seen in Table 2.1.

2.4.3 Altitude-adaptive approaches

While the methods described in 2.4.2 focus on improving recent YOLO models for UAV-based object detection, the related works in this Section aim to improve the detection by making use of the image capture altitude or more

Name	Base model	Extra Head	Loss Function	Neck	Additional Changes
SMFF-YOLO [28]	YOLOv7	✓	Wise-IoU[27]	Bi-FFP	ELAN-SW prediction head, AASPP
YOLOv7-tiny improved [26]	YOLOv7-tiny	✓	SIoU[31]	Bi-FPN	Global Attention Mechanism, Contextual Transformer
Efficient-Lightweight YOLO [11]	YOLOv5	✓	α-CIoU	ESPP	×
Drone-YOLO [30]	YOLOv8	✓	CIoU	Three-layer PAFPN, Sandwich-fusion module	RepVGG in Backbone
Efficient YOLOv7-Drone [25]	YOLOv7	✓	CIoU	×	Optimized CBS, TGM-CESC, HCEM, Removal of P5 head
PDWT-YOLO [29]	YOLOv7-tiny	✓	Wise-IoU[27]	×	Decoupled prediction heads

Table 2.1: Comparison between different publications improving the YOLO architecture specifically for UAVs.

generally speaking meta-data that is available through the UAV's sensors. Martinez-Alpiste *et al.,* proposed an altitude-adaptive approach by extending a standard machine learning algorithm to accelerate the detection speed [32]. More specifically, the authors proposed a system by first creating a customized Tiny-YOLOv3 [33] and then splitting the three output layers into three separate models.
This way the output layers that perform detection for the object sizes: large, medium, and small are matched to the different altitude levels: low, medium, and high. By splitting the model into separate detectors, the authors reduce computational complexity thus increasing inference speed.

Making use of more meta-data and labeling them as nuisances Wu *et al.,* proposed an adversarial training framework called Nuisance Disentagled Feature Transform (NDFT) [12]. In their method, the authors used data such as altitude, weather, and viewing angle that is associated with each aerial image and trained an object detector alongside a nuisance predictor. By jointly

training the detector and predictor from the same feature set in an adversarial setting, the authors developed a cross-domain object detector that is robust to UAV-specific nuisances.

In a different approach, Kiefer *et al.,* aimed to reduce domain bias by proposing a domain-aware object detector [34]. Their proposed method splits an object detection model into cross-domain and domain-specific parts to improve the performance without changing the architecture. This is achieved by first using a multi-modal dataset that includes altitude, viewing angle, and time of day as metadata and splitting it into different subsets according to the specific domains. In the next step, a chosen object detection model is pre-trained on the whole dataset, and then after pretraining concludes the weights up to a specific layer are frozen. Finally, the model will be fine-tuned on a domain-specific subset and thus become the expert on that domain.

A later work by Messmer *et al.,* introduced Adaptive Resizing, a new preprocessing step for object detection in aerial images [35]. Although limited to bird's eye view imagery, this approach achieves a significant inference speed improvement in their experiments. The main idea of the Adaptive Resizer lies in the theory that an object detector learns different representations of the same object but at different scales due to the various feeding it to the model, the size of the same object at different altitudes should result in a approximate unit size and a single representation to learn for the model.

Chapter 3

Methods

This chapter provides an overview of the research methodology and illustrates the methods used throughout this work to accomplish the outlined goal of the thesis. Section 3.1 and 3.2 highlight the type of research methods employed throughout this thesis and the steps taken for the project execution. Sections 3.3 and 3.4 describe the model and the datasets that we used in our experiments and the reasoning behind why we chose them. Then, Section 3.5 details the different altitude-adaptive methods we have worked out, while Section 3.6 explains some additional methods we have evaluated that are related to the goal of this thesis. Finally, in Section 3.7, we present the specific metrics that we employed to evaluate the performance of the different methods.

3.1 Research Method

The main research method employed in this work is quantitative research and more specifically an experimental research approach. For the majority of this work experiments were performed and validated through analysing quantitative values as a measure of how well the developed models performed. Especially when working with deep neural networks this method can be used to objectively compare the performance of two models or evaluate their training process and inference results. This kind of research method is especially relevant when dealing with the stochastic nature of deep neural networks, allowing the experiments to become reproducible, with variability limited to a standard deviation. This reproducibility is imperative for the reliability of the results, evaluation, and ablation. By selecting this research approach our study aims to contribute to a comprehensive and widely applicable understanding of the subject at hand. Although most of research performed throughout the

thesis was quantitative, at some stages we made use of qualitative research. For example, to find out what the baseline model struggled with, we qualitatively evaluated the detection results on the test images. This helped us get visual insights into which objects were missed and which were detected with an alternative method compared to using the quantitative values generated through an average of all detection results. Furthermore, when deciding where to inject the altitude information into the network, this was done based on heuristics and abductive reasoning on which layers of the network are associated with which functions since time and resource constraints would not allow performing hundreds of experiments to see which layer would be ideal.

3.2 Research Process

The extensive literature review about state-of-the-art object detectors was essential for understanding the process of how these networks learn to detect objects in images and the reasons for the challenges they still face. Performing data analysis to comprehend the appearance of images taken by a UAV warranted a further literature review more focused on altitude-aware object detection models. Among the methods found, none of them featured a learning-based approach, instead they mostly focused on using either different models for different altitudes [32] [34] or preprocessing techniques [35]. With the core idea of altitude data being relevant still validated through their research and the understanding of how YOLOv7-tiny works, this served as a reason to research dynamic or learning-based approaches. Consequently, the ensuing research approach can be divided into the following steps:

- Model selection (1)
- Dataset selection (2)
- Experimental phase
  - Architectural changes (3)
  - Evaluation of performance (4)
  - Repeat step 3 and 4 (5)
- Analysis of detection performance improvement or deterioration (6)

Figure 3.1 shows a schematic representation of this approach prior to the final analysis.

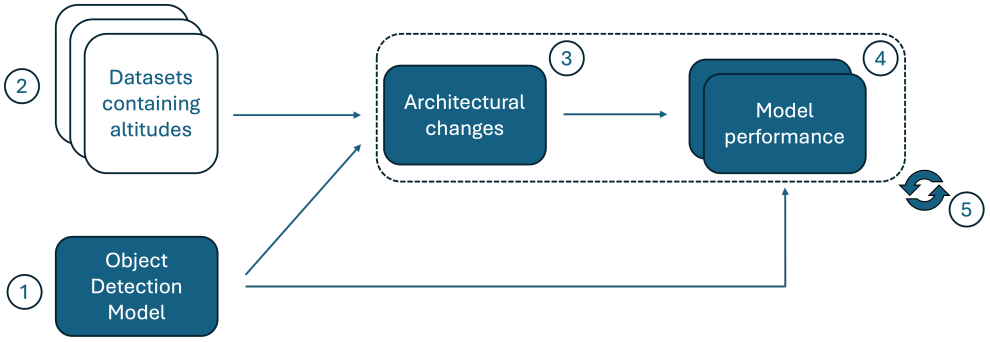


Figure 3.1: Illustration of research process: 1) Identification and selection of baseline model 2) Selection of altitude containing datasets 3) Change the underlying model architecture to make use of altitude 4) Evaluate performance of modified network and compare to baseline performance 5) Repeat step 3 and 4.

3.3 Model Selection

The selection process for the model considered the following key factors: novelty, benchmark performance for real-time object detection, number of parameters, and acceptance for UAV-based applications. To ensure the relevance of the study, novelty was prioritized. Given that a large portion of the studies in the related work section 2.4 build upon the same model or a variation thereof and have been published less than a year before the start of this work, this highlights the novelty and benchmark performance of the YOLOv7-tiny model selected. Secondly, in the broader application space of UAV-based object detection, real-time inference speed, and model size, the chosen model performs well enough to be still considered state-of-the-art. All the mentioned criteria further enhance the generalizability and relevance of this study.

YOLOv7 provides noteworthy improvements in detection accuracy and computational efficiency as one of the latest models in the YOLO series. The published YOLOv7 algorithm comes in several versions based on the model's depth and width but they share the same general structure. With a good balance between precision, model size, and detection speed, the smallest version: YOLOv7-tiny proves an optimal choice for this aerial drone-based object detection study.

3.4 Datasets

In recent years the amount of datasets captured by UAVs that were published has risen. UAVDT [36] and VisDrone [9] are some of the more notable ones encountered in literature and similar to several other UAV-based datasets, they depict traffic situations. Other datasets that have been made publicly available such as DOTA [37] for cities and BIRDSAI [38] for wildlife and humans datasets have been taken by satellites and thus feature very high altitude images but without the variation in altitude and angle a UAV-based dataset would have. Similarly to the BIRDSAI dataset, the HIT-UAV [39] dataset features thermal infrared images compared to the usual RGB images featured in the majority of UAV and satellite datasets.

3.4.1 Multi-modal Datasets

HIT-UAV is one of the more recent datasets that acknowledged the potential of metadata captured by the UAV providing this essential flight data for each image. The dataset comprises 2,898 infrared images in various scenes mainly featuring people, cars, and bicycles. The People On Grass dataset [34], while slightly older than HIT-UAV, is also one of the few publicly available datasets we found that included metadata such as the altitude and camera angle, among others. The 2900 images taken by a drone contain people in the same scene but on a variety of angles ranging from 0° to 90° and altitudes ranging from 5m to 110m. The metadata included in the UAVDT is not as precise as the two previously mentioned datasets. Nevertheless, for each image, it is possible to know at which of the three altitude levels — low, medium, or high — it was taken and whether the camera view angle can be described as front-view, side-view, or bird-view. The dataset captured by FLOX depicts thermal infrared images taken by a drone flying over various fields in Sweden and has manually annotated labels for the wildlife that could be identified. The classes of animals in this dataset include 'boars', 'deers', and the generic class 'animal' if the animal could not be identified by the human annotator which primarily occurs at high altitudes. However, only a small percentage of the images in this dataset contain annotations for the capture altitude and the camera angle is missing entirely.

See Table 3.1 for a summarized overview and comparison of the multi-modal datasets described in this Section with sample images shown in Figure 3.2.

Name	Objects	Altitude	Angle	Type	#Images
UAVDT (BV)	Vehicles	5-200m*	70-90°†	RGB	4824
HIT-UAV	Vehicles& People	60-130m	30-90°	Thermal	2866
POG	People	5-110m	0-90°	RGB	2892
FLOX	Animals	5-60m	0-90°	Thermal	2987‡

Table 3.1: Comparison between the datasets used in experiments over the course of this work. For the UAVDT dataset we only used the images labeled as bird-view and for the FLOX dataset only a small subset included the altitude information.

While all the datasets summarized in Table 3.1 were used throughout the experiments of the thesis since they satisfy the requirements of being taken by a UAV and including altitude information, the main results are reported on the PeopleOnGrass (POG) dataset. The reason we chose to use this dataset to report the results is because it is well-balanced regarding different parameters compared to the other datasets, making its results more meaningful. The HIT-UAV only has images taken from 60 meters and above, which makes its lowest altitude images correspond to medium and high altitude images in the other datasets. Additionally, the object size differences are not as large in a range of 60 to 130 as opposed to 0 to 60 for example. The UAVDT dataset on the other hand has a wider range of altitudes but lacks granularity on its altitude values. While most experiments in this work have been performed on discrete altitudes, future research could investigate more or even continuous altitude levels. Furthermore, vehicles at high altitudes still occupy significantly more pixels than people or animals, which makes it not as suitable when investigating the detection of smaller objects. The FLOX dataset is relevant for the use case of wildlife detection and is a challenging dataset where the baseline model struggles significantly at detecting objects at high altitudes. However, only less than 200 images have their altitude annotated making it a very small dataset, thus limiting its meaningfulness. While only being a single-class dataset the POG dataset offers good comparability through its simplicity and its similarity in one or another aspect to the other datasets.

*Annotation is not exact but marked as low, medium or high altitude

†UAVDT dataset contains angles labeled front-, side-, bird-view with bird-eye view corresponding to an angle of around 90°

‡Only 175 out of 2987 included altitude information

(a) (b) (c) (d) (e) (f)

Figure 3.2: Comparison of six sample images from the datasets used in this work. (a) is a sample from the UAVDT dataset with cars and trucks from medium altitude and bird-view perspective. (b) is a sample from the POG dataset with people at an altitude of 45m and a camera angle of 90°. (c) is a sample from the HIT-UAV dataset with people and cars at an altitude of 120m and a camera angle of 40°. (d) is a sample from the FLOX dataset with deer at an altitude of 40m and a camera angle of 30°. (e) and (f) are low alitude images from POG and FLOX respectively with annotated bounding boxes for reference.

3.5 Altitude-Adaptive Object Detection

After a thorough analysis of the YOLOv7-tiny architecture and related work, we identified multiple methods on how to include the altitude information into the training process of the network. The methods can be categorized

into two dynamic approaches (3.5.1, 3.5.2) and three learning-based methods (3.5.3, 3.5.4, 3.5.5). While the dynamic approaches make use of the altitude knowledge to dynamically adapt parameters of the network leaving the architecture itself untouched, the learning-based methods are based on an adaptation of the original YOLOv7-tiny architecture and use the altitude information as input or labels.

3.5.1 Dynamic Anchors

As an anchor-based model, YOLOv7-tiny relies on a good choice of anchor configuration. The default anchor configuration likely works well for datasets that are similar to the COCO dataset or other natural image benchmark datasets. When working with UAV-datasets it helps to use an anchor configuration where the anchors are slightly smaller for all three prediction head layers. With the prediction heads relying on a good choice of anchors for bounding box regression and thus detection performance it is vital to find a fitting configuration for the dataset.

Since the set of anchors is dependent on the dataset used and even then a possible hyperparameter tuning process, we used the auto-anchor function provided by YOLOv7-framework for the dynamic anchor experiment 3.5.1. This function clusters the objects in the dataset, using the centroids of the clusters to provide an initial anchor configuration, which is then optimized further using a genetic algorithm. This method gives us the possibility to find a consistent way to get a good anchor configuration for a given dataset or subset without any manual tuning involved. As we are only interested in the effect of choosing a given set of anchor configurations based on the altitude, the absolute best configuration does not matter. This algorithmic approach also allows the experiments to be simpler to reproduce and more generalizable.

To get good anchor configurations for different altitudes, the datasets were split into subsets according to their altitude into low, medium, and high altitude subsets. This way the auto-anchor function could be run on each specific subset as well as the full dataset, generating an anchor configuration for each one. Given these configurations for the different altitude levels, we can then compare their performance for each altitude level and modify the object detection model to use a given set of anchors for a given image based on that image's altitude.

While these findings resulted in different anchors for different altitudes, our goal was to design a general object detector that performs better overall. For that reason, we used the previously identified anchors in a dynamic

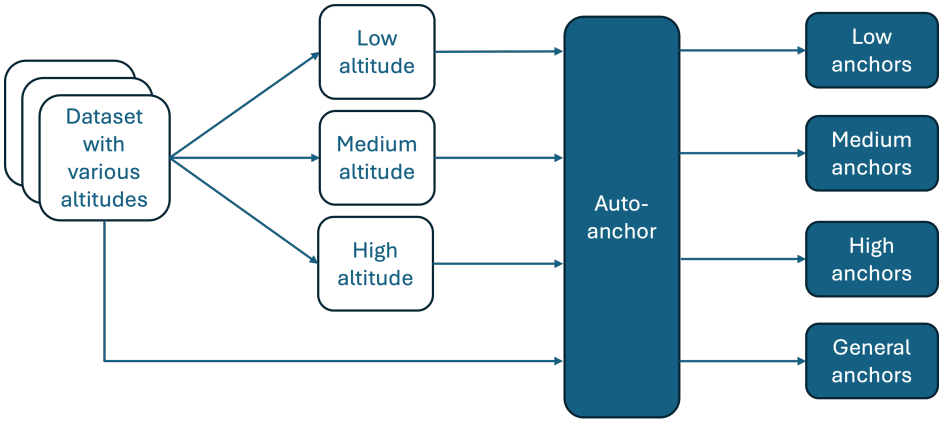


Figure 3.3: Process of generating the anchor configurations for the different altitudes.

approach. This was done by including all the domain-specific anchors as hyperparameters in the configuration file and then while the model trains, it would choose the anchor set configuration that corresponds to the altitude the training image was captured at.

3.5.2 Dynamic Loss

Through empiric testing, we found out that one of the reasons the detection performance was still lacking was the recall. If the recall is not at 1.0 at a confidence level of 0 then the model has not been able to detect all instances of objects, since the objectness score predicts if any grid cell of the feature tensor contains an object or not. Consequently, if we prioritize the objectness term in the loss function we push the model to pay more attention to missed detections. If we assume that the model misses more objects at high altitudes because the objects are smaller and the exact location of the bounding box is not as important, then increasing the objectness loss gain at high altitudes could lead to the intended behavior of improving recall and thus improving object detection overall without obstructing the performance at lower altitude.

The dynamic objectness loss gain γ_{dyn} can be described through the original gain γ_{obj} and a new hyperparameter α:

$$\gamma_{dyn}(\alpha, \gamma_{obj}) = \begin{cases} \alpha \cdot \gamma_{obj} & \text{if altitude} = \text{high} \\ \gamma_{obj} & \text{if altitude} \ < \text{high} \end{cases}$$

Note that when $\alpha > 1$ the objectness loss increases for higher altitudes and when $\alpha < 1$ it decreases.

In the actual implementation, the images are loaded as a batch. This means that not all images in a batch have are necessarily of the same altitude, however, our analysis of the data loading showed that most of the images in a given batch were still of the same altitude. This being the case and since objectness loss is calculated for the whole batch at once we decided that at least 80% of the images in a batch have to be of high altitude for the dynamic loss to activate. In other words, if less than 80% of the images in a given batch were of high altitude, then the original loss object gain γ_{obj} would be used.

3.5.3 Altitude Injection

A third alternative to include altitude information into the object detection process and have the model learn from it was to give the altitude as input to the model. The reason for this alternative was multi-fold. If the altitude information holds additional value to the model it is more free to learn any relationship between learning object detection without being limited to a handcrafted choice that only influences anchors or objectness loss. A hypothetical correlation it could learn is weighing the predictions of the prediction heads differently depending on the altitude input. For example for high altitudes with a greater number of small objects, the shallower head tends to hold more weight. With only one set of anchor configurations, the altitude input could help the model learn faster which anchors are better at which altitudes. It could also increase the confidence for object detection as it is more likely that certain object classes or sizes are at certain altitudes. This altitude injection approach can range from a very naive approach to more intricate approaches that take the model architecture into account. In this Section, we will describe all the different ways altitude was given as additional input to the model and which architectural changes were performed.

By analyzing three methods of Altitude Injection we were able to get insights into where such an extra input injection might be more useful or more disturbing to the learning process of the YOLOv7-tiny model.

Altitude Injection at Input

As mentioned before, there are multiple ways to inject the altitude information as input into the model. The simplest way to do that is to concatenate the input image with the altitude information as shown in Figure 3.4. As the altitude information is just a single value, we needed to copy and expand it to a tensor of the same dimension as the input image. We chose this method as it gives the model the information at the earliest time possible and thus more freedom

and time to propagate it. However, if the information is only relevant to the network close to the output layers it might be harder for it to learn a correlation.

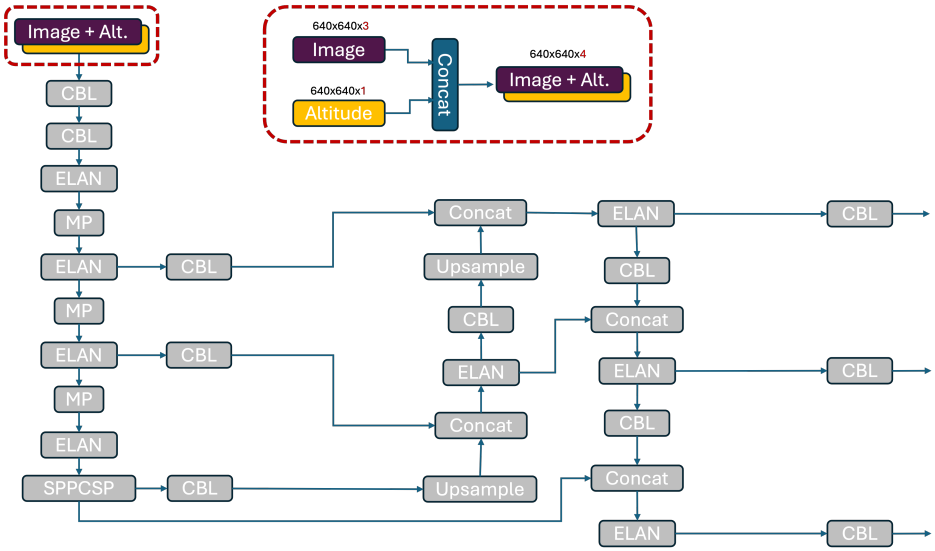


Figure 3.4: YOLOv7-tiny architecture with the original input adapted to use the image concatenated with the altitude as input.

Altitude Injection at Neck

With the backbone layers being responsible for feature extraction, the dimension of the input being very large, and the altitude not holding any pixel-related information it might make the feature extraction harder if the information is injected at the input layer. As described in the background 2.3, the FPN is important for fusing the features of different scales. As an attempt to inject the altitude closer to these feature fusion layers so that the altitude information can influence how important *i.e.,* how much weight the different feature maps of each depth should be attributed to, we injected the altitude information after the backbone and before the FPN layers as can be seen in 3.5. The idea is, that the model learns to weigh deep features less at the concatenation and for further propagation when the capture altitude of the input image is high and the objects are smaller and conversely weigh them more when the capture altitude is low.

For the model to learn directly from the altitude input, an additional 1x1 convolution is added after the layer where the information is injected. This means at the injection layer, the altitude tensor is concatenated with the output of the previous layer. This creates a feature map with one extra channel. To

reduce the channel by one again to fit the next layers' dimensionality a 1x1 convolution is added which is supposed to integrate the altitude information with the other features.

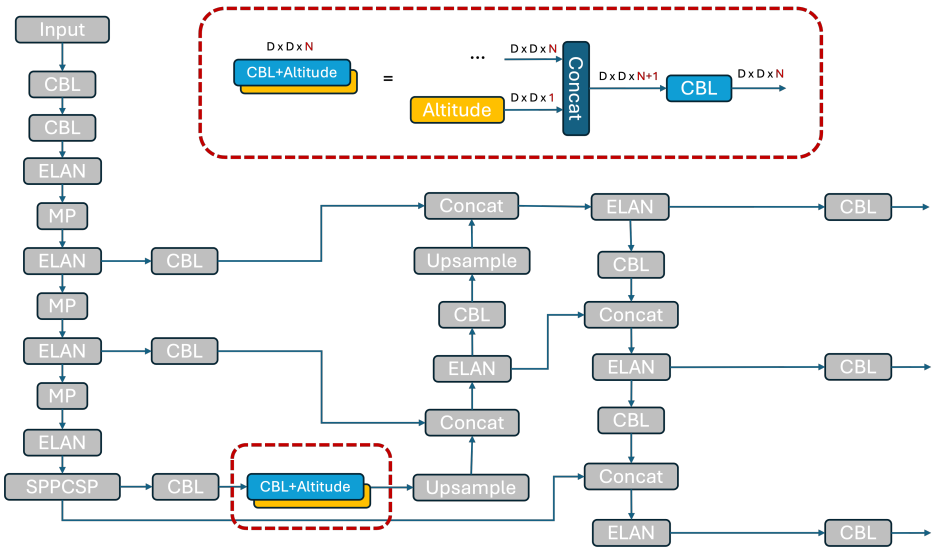


Figure 3.5: YOLOv7-tiny architecture with altitude tensor concatenated to the feature map coming out of the backbone and an additional 1x1 convolutional layer to keep original dimensionality.

Altitude Injection at Prediction Head

The third and final position to inject the altitude information into the network investigated in this thesis was after the neck and before the final prediction heads. This can be seen in Figure 3.6. The advantages of this method are that the altitude information is not lost through multiple convolutions and the additional input cannot negatively impact the feature extraction and feature fusion of the network. If the altitude information is only relevant to the network for deciding which prediction head and which anchors to assign more confidence to, then it can be helpful that the relevant information is as close to the prediction and anchor layer as possible. Furthermore, if we assume that additional non-image input can confuse the model feature extraction and feature fusion process or is treated as noise, then adding the altitude at the end should not influence the earlier processes. On the other hand, the previous layers will be oblivious to this additional input and there will be significantly fewer weights that can learn from it.

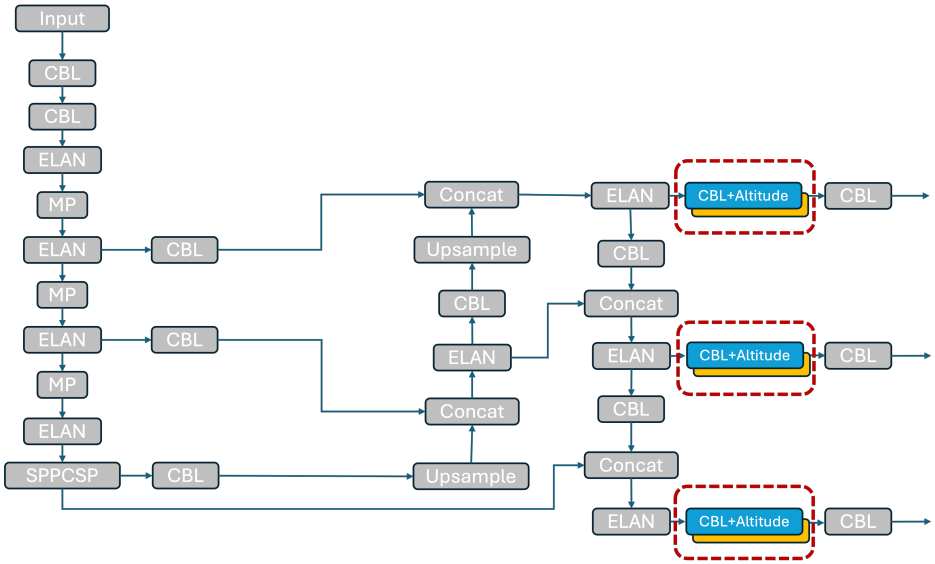


Figure 3.6: YOLOv7-tiny architecture with an additional 1x1 convolutional layer concatenated by the altitude tensor before each prediction head.

3.5.4 Altitude Prediction

As an alternative to giving the altitude as an input, we also investigated using the altitude as an output label. This allowed us to observe whether the altitude information is embedded into the image in a way the object detection model can learn it. If the model could perfectly predict the altitude from the image alone, it might be less relevant to use the altitude information as additional input.

The method we used to implement this altitude prediction was through the addition of an auxiliary altitude prediction head for each object detection prediction head (see Figure 3.7). This auxiliary head was implemented similarly to the classification part of the object detection head where the labels are one-hot encoded and thus the network will predict the probability for each altitude level and the optimization is done through a binary cross-entropy loss. Finally, this additional loss for the altitude will be multiplied by a tunable hyperparameter gain γ_{alt} and summed up with the existing losses:

$$\mathcal{L}_{total} = \gamma_{box} \cdot \mathcal{L}_{box} + \gamma_{obj} \cdot \mathcal{L}_{obj} + \gamma_{cls} \cdot \mathcal{L}_{cls} + \gamma_{alt} \cdot \mathcal{L}_{alt} \tag{3.1}$$

The reason we chose to implement it this way was because on one hand the earlier experiments were performed with discrete altitude levels and on the other hand it also should be easier for a network to learn binary cross-entropy

rather than a mean squared error [40]. Furthermore, this removes the need for the altitude values to be precise, which is not necessarily a given, and thus lessens possible variance.

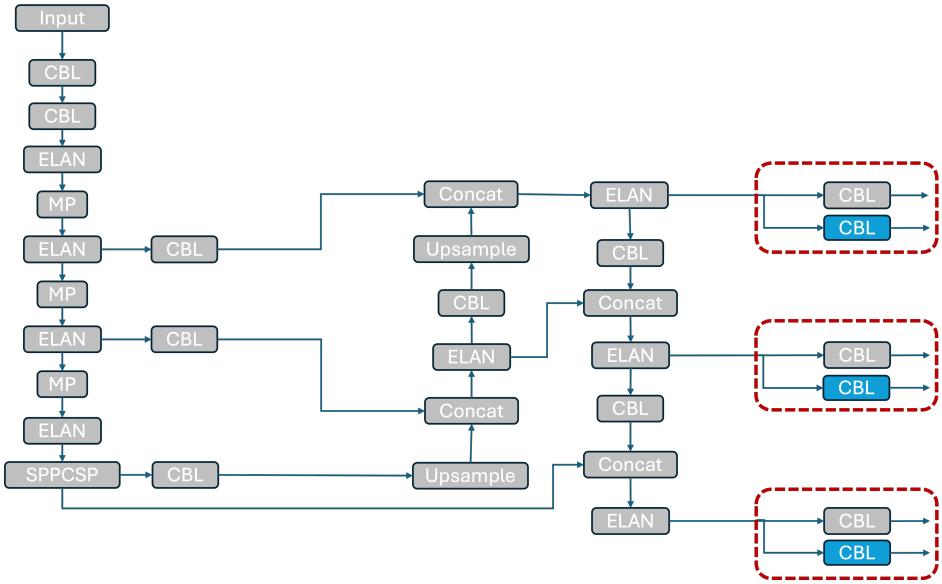


Figure 3.7: YOLOv7-tiny architecture with auxiliary altitude prediction head.

3.5.5 Combination of Altitude Injection and Prediction

The final method to train the model on the altitude information researched throughout this thesis was a combination of the altitude injection 3.5.3 and the altitude prediction 3.5.4 methods. The idea behind this approach was to incentivize the model to learn from the injected altitude information instead of solely relying on the image. The altitude prediction experiments showed that the object detection model could not fully predict the correct altitude information from the input alone. On the other hand, the altitude injection method assumed that the model can learn a correlation between the altitude information and object detection by itself and would not disregard it as noise. Since this seemed to be the case in some of the experiments we proposed our combinatory approach which would force the model to propagate the additional information from the injection layer up to the prediction heads of the model. This way we made sure that the altitude information would not get lost in the forward propagation and was present at any layer after the injection point where it could be relevant. Furthermore, this approach provided us with

further insights into if and when during the training process the model learns the correlation between the injected altitude and the altitude prediction and if that leads to a noticeable increase in performance.

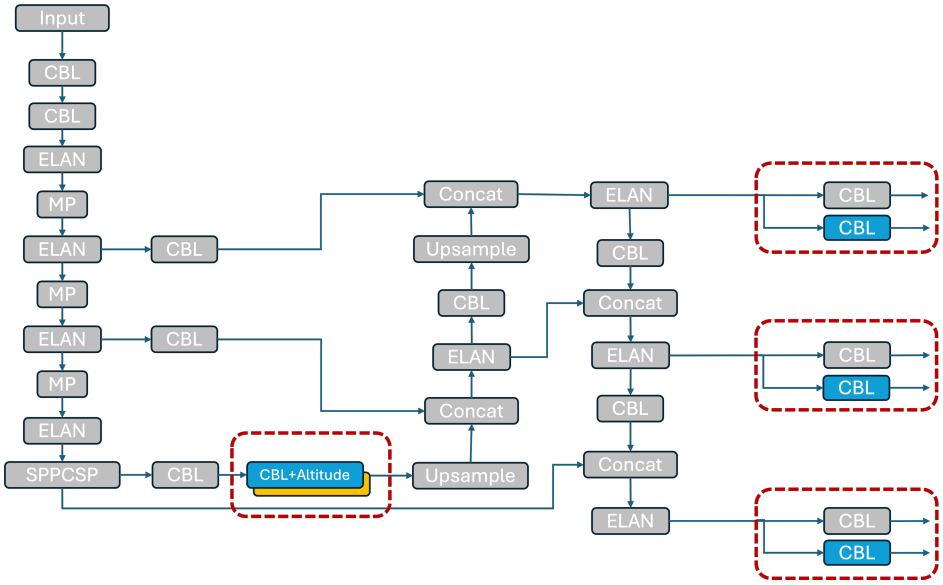


Figure 3.8: YOLOv7-tiny architecture with auxiliary altitude prediction head and injection layer at the neck.

3.6 Exploratory Analysis

The following Section describes the methodology used for additional experiments that tackle the same challenge of small object detection for aerial images but through different methods. These methods do not focus on a learning-based or dynamic approach using altitude information but rather investigate how important some layers of the network are for aerial images or investigate a separate model optimized for UAVs on our datasets to validate the original author's findings.

3.6.1 Removing the FPN layers

Training a model with the FPN layers removed was used to quantify how important the multi-scale features of the object detection models are for UAV images. According to what we detailed in 2 when the camera angle of the UAV is close to 90 degrees or at sufficient enough altitudes, there is little

to no scale variation in the objects of a given image. This in turn would hypothetically remove the main need for the feature pyramid network which is to share features across different depths and scales of the network which are relevant for detecting different sizes of objects.

The removal of the FPN part from the YOLOv7-tiny model is illustrated in Figure 3.9. Compared to the original YOLOv7-tiny architecture 2.1, the vertical connections in the center of the network are removed. This way the three different prediction heads do not share their features. While this method was not implemented in the hopes of improving detection performance, the removal of the middle layers reduced the number of parameters and thus the size and complexity of the model. With this approach, we could gain further insights into the detection performance drop to memory and speed improvement trade-off.

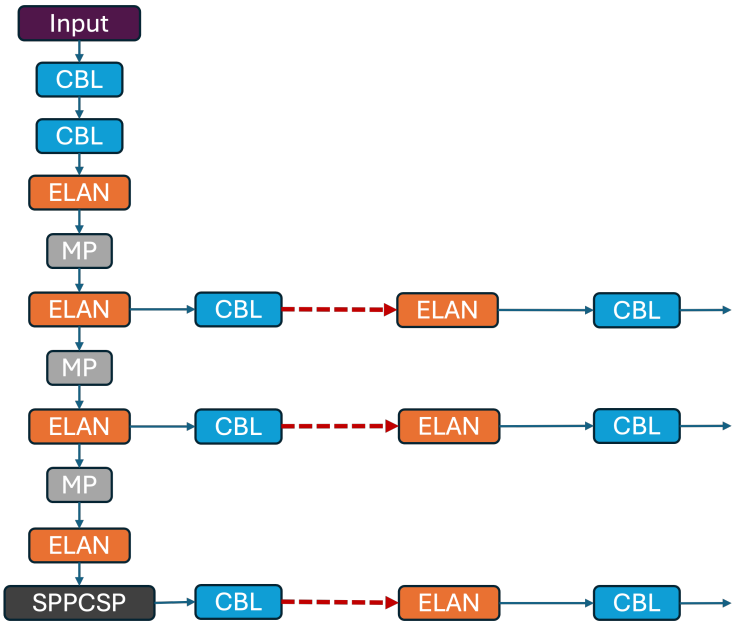


Figure 3.9: YOLOv7-tiny architecture with the FPN part of the network removed.

3.6.2 Multi-experts

To further validate that the YOLOv7-tiny model could benefit from the altitude meta-data, we performed an experiment similar to Kiefer *et al.* [34]. First, a general YOLOv7-tiny model was trained on the whole dataset for enough epochs that the validation loss converges. The earlier layers of

this trained model were frozen and then, on each altitude-level or domain respectively a new model was further trained on the subset of the whole dataset corresponding to that domain shared by the domain-specific layers. This effectively fine-tuned different models on different altitude levels with the weights of its earlier layers shared. The shared layers were the backbone plus the Spatial Pyramid Pooling layer, that is to say, the first 37 layers for our YOLOv7-tiny model. While already investigated in previous research, the authors used different models, namely DE-FPN [9] and EfficientDet [41]. Doing this experiment allowed us to gain additional knowledge and compare this method to the other approaches investigated in this thesis.

3.6.3 Additional Head

Our final exploratory analysis investigated the matter of improving small object detection as a whole for aerial images without necessarily incorporating altitude information. As mentioned in Section 2.4.2 multiple publications were promising UAV-optimized YOLO models. What all the proposed methods had in common was the use of a small object prediction head, conventionally named P2, either as an addition or instead of the largest object prediction head P5. Thus, we adapted the YOLOv7-tiny architecture according to Figure 3.10 to our P2-YOLO.

While the related works proposed more changes than only the additional prediction head we wanted to explore how and if this change alone would improve the detection performance on the datasets we had available. We evaluated and compared our adaption with the performance of the PDWT-YOLO proposed by Wang *et al.* [26]. The reason we compared against this model was that it was the most recent one out of the described publications and that it was built upon YOLOv7-tiny which is the baseline model used throughout the thesis.

While simply incorporating one of the changes mentioned in multiple publications or reusing the PDWT-YOLO cannot be considered novel by itself, analyzing their performance on all our datasets is important in validating the published findings and it allowed us to compare the methods with themselves, and to the other methods we investigated.

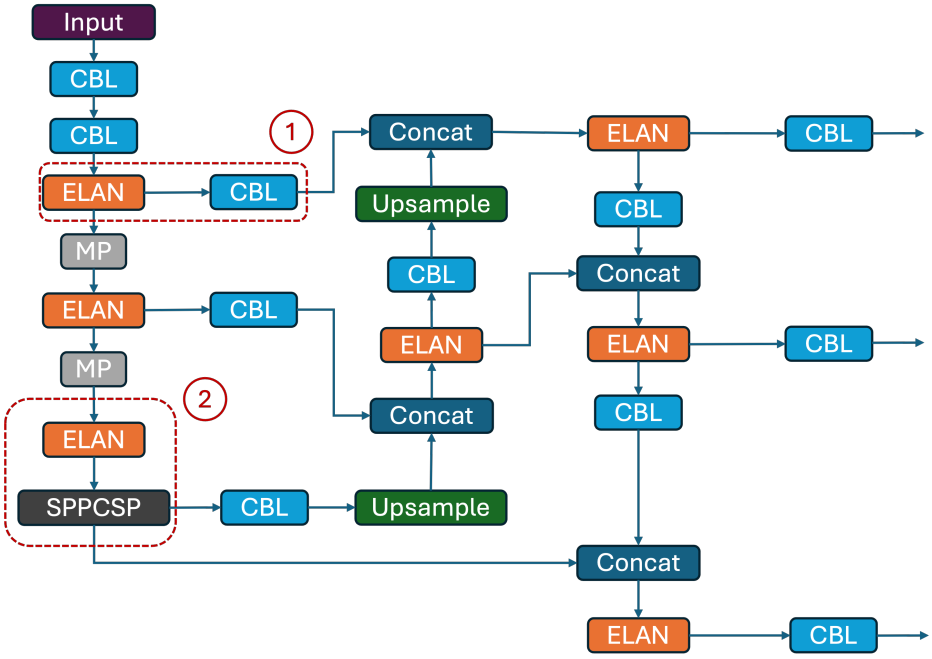


Figure 3.10: Architecture of P2-YOLO, an adapted YOLOv7-tiny that has 1) the first head connected to the P2 feature map and 2) the P5 feature map and all its connections removed but instead used the SPPCSP layer at P4.

3.7 Performance Evaluation

We conducted different experiments to show whether or not incorporating altitude information could improve the performance of the baseline object detector. Then we conducted further ablation studies to examine how the different methods relate to each other and which factors influence the model most. This is also the reason why we performed this variety of experiments with multiple approaches to obtain knowledge about existing and novel approaches and discuss which approaches might still hold value and which can likely be ignored for future research.

The evaluation of the detection performance of the different models is made using some of the most common benchmark metrics for object detection, namely Precision, Recall, and Mean Average Precision (mAP). While precision and recall are reported in our results for better interpretability one of the key metrics when comparing detection performance is the mAP_{50} metric. As a standard for benchmarking in object detection and the official Pascal-VOC metric [42] it allows for a comprehensive evaluation by taking into account both precision and recall at a threshold of 50% IoU and combining

them into a single performance score. To get additional insights into the models' performances under varying levels of detection strictness we include the mAP metric as defined in the COCO paper [2] which averages the mAP values over IoU thresholds ranging from 0.5 to 0.95 at an interval of 0.05. In this thesis, without any subscript mAP or AP refers to this metric.

In addition to precision, recall, and mAP, other metrics such as the number of model parameters or Floating Point Operations (FLOPS) can help evaluate and compare different models. Although only to a limited extent, these parameters can provide insights into the model complexity and thus memory footprint and inference speed.

The following subsections give a brief description of how the metrics are computed.

3.7.1 Precision and Recall

Precision is the ratio of correct prediction over the total number of positive predictions and thus measures how accurately a model detects positive instances:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{3.2}$$

On the other hand, recall is the ratio of correct predictions over the total number of ground-truth positives and thus measures how well a model can detect all relevant instances:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3.3}$$

Here, TP (true positives) refers to the number of correctly detected objects, FP (false positives) to the number of incorrectly detected objects and FN (false negatives) represents the number of objects the detection model has missed.

Positive and negative detections are defined according to the IoU detailed in 2.3.1. The IoU defines the threshold between what is considered a positive and a negative prediction. For mAP_{50} for example, this threshold is at 50%, so everything above that threshold is a positive prediction and everything below a negative prediction.

3.7.2 Mean Average Precision

Average Precision (AP) is calculated for each class individually and summarises the shape of the precision-over-recall curve. It approximates the area under the curve by using the 11-point interpolation method, that is the mean precision at eleven equidistant recall levels:

$$\text{AP}_{interpol} = \frac{1}{11} \sum_{r \in \{0.0, 0.1, 0.2, \ldots, 1.0\}} P_{\text{interp}}(r)$$

with P_{interp} being the interpolated precision at recall r defined as:

$$P_{\text{interp}}(r) = \max_{\tilde{r} \geq r} P(\tilde{r})$$

Finally, the mAP is, as the name implies, the mean of the average precision of each class:

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{3.4}$$

Note that for single-class datasets such as POG, mAP is equivalent to AP.

3.7.3 Accuracy

While not used to evaluate the detection performance of our models, we use the accuracy metric to evaluate whether the models containing an auxiliary head correctly predicted the altitude of an image. We define accuracy as follows:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \tag{3.5}$$

3.7.4 FLOPS

While the exact calculation depends on the specific kind of layer, YOLOv7-tiny mainly consists of convolutional layers where the FLOPS are calculated as follows:

$$\text{FLOPS} = 2 \times k \times k \times n \times w \times h \times m, \tag{3.6}$$

where $k =$ filters, $n =$ input channels, $w =$ output width, $h =$ output height, and $m =$ output channels.

The total FLOPS can be calculated by calculating the FLOPS of each layer of the model and then summing it up.

$$\text{FLOPS}_{total} = \sum_{i=0}^{N} \text{FLOPS}_i \tag{3.7}$$

with $N =$ Number of layers and $\text{FLOPS}_i =$ FLOPS of layer i.

Chapter 4

Results and Analysis

In this chapter, we will present the results that we obtained from our main experiment. Initially, we will describe the results of our data analysis in Section 4.1 which serve as a foundation for the methods proposed in the previous chapter. Then we will briefly describe the experimental setup relevant for evaluating our finding in Section 4.2. The main results of the altitude-adaptive methods are presented in Section 4.3 while the additional results are described in Section 4.4. Finally, we will compare the best results of each separate method in a cross-method evaluation in Section 4.5.

4.1 Data analysis

To validate the assumption that all the objects belonging to the same class have a similar size at certain altitude levels we performed some data visualization. For that, all the images that were captured in between a range of altitudes were discretized to three different levels according to the full range of altitudes of a given dataset. Altitude values belonging to the lowest third are classified as ”low altitude”, values in the range of the next third are ”medium altitude” and ”high altitude” for the last third. For the POG dataset, this corresponds to 5m to 40m for low altitude, 40m to 75m for medium altitude and 75m to 110m for high altitude. As can be seen in Figure 4.1 there is a distinction between the object size of a single class, described by its area, and the altitude level. While there is some overlap in the curves, it is clear that with increasing altitude, the majority of the objects are smaller in size than at lower levels. The same was true for the other three datasets examined: UAVDT, HIT-UAV, and FLOX. It is important to note, that part of the overlap can be explained by the range in capturing angles and the accuracy of the altitude levels. For example, a

shallower angle can cause great scale variations at lower altitudes resulting in smaller objects in the background and some high ”medium altitude” images can be similar to low ”high altitude” images in terms of object size.

To ensure that the results are not skewed because of imbalances in the datasets, we processed them first to guarantee a uniform distribution over the different altitude levels if the dataset was not already balanced from the start. Thus, in all our experiments the training, validation, and test set contained approximately the same number of images for each altitude level.

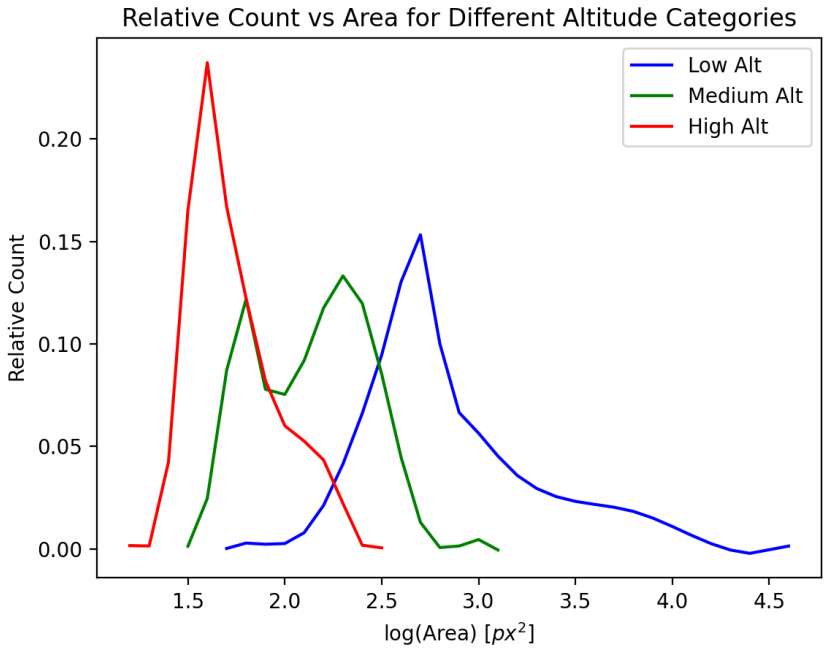


Figure 4.1: The relative number of objects with a certain log area for three discrete altitude levels for POG which ranges from 5m to 110m. Since POG is a single class dataset the three distinct curves show the correlation between altitude and object size of the same class *e.g.,* the mean object size of the people in the image at high altitude is smaller than at medium altitude which in turn is smaller than at low altitude.

When plotting the height and width of the individual objects for the images of the whole dataset and the specific altitude levels we can not only get insights about their size but also their aspect ratio. If we then overlay these plots with the anchors we found through the method described in 3.5.1 we can see the results in Figure 4.2. We can observe that the general anchors that were generated from the full dataset do not fit as well on the objects at high altitudes as the anchors generated from the high altitude subset. However,

when regarding all objects of a dataset we can easily see that the more specific high anchors only capture a small number of objects well, while the general anchors span across more objects. Similar observations could be made for the other anchors on the other altitude levels.

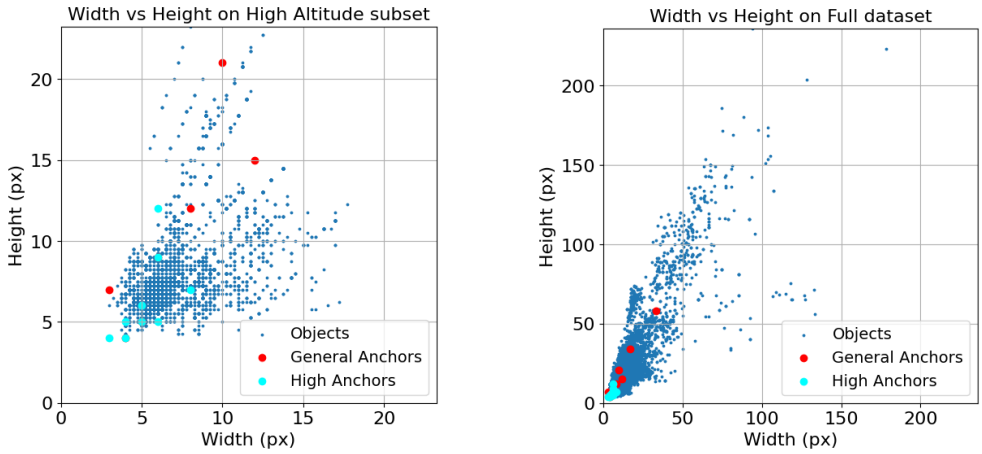


Figure 4.2: Example of how well general anchors fit compared to altitude specific anchors on the High Altitude subset (a) and the full dataset (b) of POG.

4.2 Experimental Setup

While we performed the initial experiments with a variety of training epochs and over multiple datasets, the final experiments reported in this section have been mainly performed on the POG dataset and have been trained from scratch for 500 epochs with a batch size of 32. To get an estimate for the standard deviation, we trained the baseline model three times and reported a standard deviation of 0.2. With this in mind, for all the following experiments the models have been trained only once due to time and resource constraints. Furthermore, unless otherwise stated, the models were trained with the default parameters found in the official YOLOv7-tiny repository [43] except for the 'loss_ota' parameter which we set to 0 for faster training and better performance of the YOLOv7-tiny baseline. During training, we saved the best model of all the epochs which is defined by a weighted sum of the mAP_{50} and mAP on the validation set, where mAP is given nine times the weight of mAP_{50} . Then this best model is evaluated on the test set which stems data from the same distribution but was never seen by the model during training. As mentioned in Section 3.7, without any subscript (m)AP refers to the average value over thresholds from 0.5 to 0.95 in steps of 0.05.

4.3 Altitude-Adaptive Object Detection

This Section reports the result of the altitude-adaptive approaches described in 3.5. It consists of experiments using dynamic anchors (4.3.1), a dynamic loss function (4.3.2) and using altitude as input (4.3.3, as labels 4.3.4) or a combination of both (4.3.5).

4.3.1 Dynamic Anchors

For the initial part of this experiment, two sets of anchors were chosen with which the models were trained on the HIT-UAV dataset and tested on two subsets of the test set. One subset only included images at an altitude of 60 meters and the second subset only included images at an altitude of 130 meters. The experiment showed that at lower altitudes, when the objects were larger, the anchor set that contained larger anchors performed better, and at higher altitudes the opposite was true and the smaller anchors performed better.

As described in Section 2.4.1 the follow-up experiment integrated the two sets of anchors into the model. After training the model, however, it was clear to see that it could not learn as effectively as the baseline. Indeed, the training objectness loss increased even initially. Even after decreasing again the objectness loss stopped decreasing similar to the validation box loss. This was accompanied by a loss in average precision as seen in Figure 4.3.

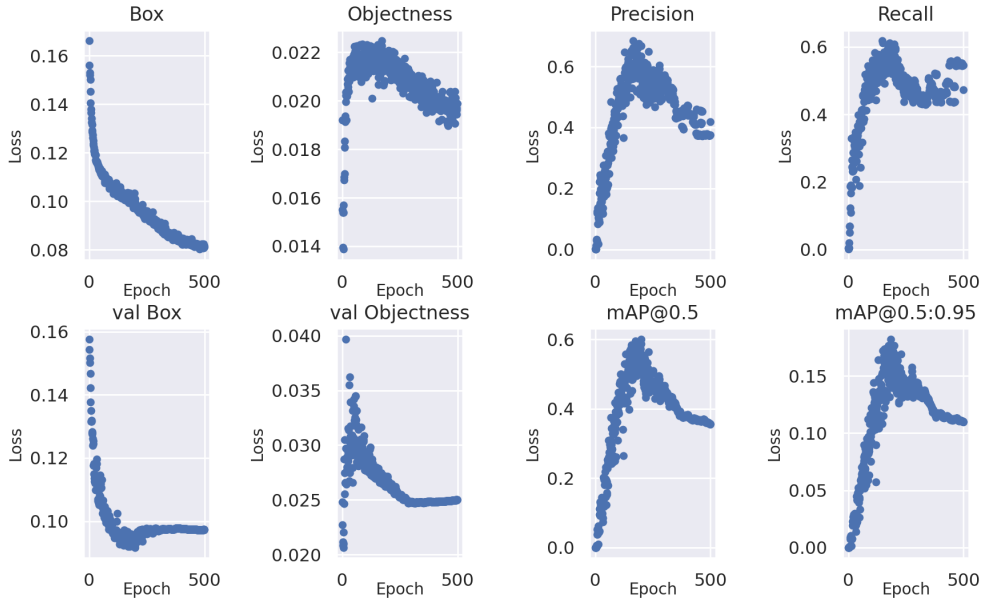


Figure 4.3: Results of the training process with the dynamic anchors.

Investigating the findings further, we trained the model with one set of anchors like the baseline model, but instead included all the anchors into this larger set. This means the anchor set consisted of 3×9 anchors instead of the original 3×3. The results showed that this also resulted in a performance drop compared to the baseline.

4.3.2 Dynamic Loss

For the dynamic loss, *Baseline A* is the default YOLOv7-tiny model which has an objectness loss gain of $\gamma_{obj} = 1$. For a fair comparison and further insights, we trained another baseline model, *Baseline B*, with a gain of $\gamma_{obj} = 1.5$. We evaluated three different dynamic loss models with different gains and values for α. Namely *Dynamic A* with $\gamma_{dyn}(\alpha = 1.5, \gamma_{obj} = 1)$, *Dynamic B* with $\gamma_{dyn}(\alpha = 1.33, \gamma_{obj} = 1.5)$ and *Dynamic C* with $\gamma_{dyn}(\alpha = 0.67, \gamma_{obj} = 1.5)$. The resulting performance of these models can be seen in Table 4.1. Note that for $\alpha = 1.5$ and $\alpha = 1.33$ the objectness loss gain was increased at high altitudes compared to the specified γ_{obj} and for $\alpha = 0.67$ it was decreased.

We can observe that the baseline model with the higher gain performs better or equal in AP to the different dynamic losses in most instances except the medium altitude level while the highest recall for higher altitude images was achieved when the dynamic objectness loss gain was smaller at high altitudes.

Full	*P*	*R*	AP_{50}	*AP*
Baseline A	89.5	78.2	84.5	42.5
Baseline B	**91.0**	77.7	**85.3**	**43.6**
Dynamic A	89.9	**79.0**	**85.3**	43.0
Dynamic B	90.1	78.1	84.1	43.5
Dynamic C	90.4	**79.0**	85.2	42.8

Medium	*P*	*R*	AP_{50}	*AP*
Baseline A	90.1	83.0	88.7	43.2
Baseline B	93.2	80.9	89.1	43.9
Dynamic A	**93.4**	83.2	89.2	42.5
Dynamic B	91.2	81.3	87.6	**44.3**
Dynamic C	91.1	**84.2**	**89.6**	43.0

Low	*P*	*R*	AP_{50}	*AP*
Baseline A	**97.8**	87.2	93.5	54.0
Baseline B	95.5	90.1	94.9	**55.9**
Dynamic A	95.5	90.3	**95.0**	54.7
Dynamic B	95.8	**91.6**	94.3	55.1
Dynamic C	96.1	88.9	94.1	54.4

High	*P*	*R*	AP_{50}	*AP*
Baseline A	84.5	68.0	73.9	34.7
Baseline B	84.6	67.6	**74.8**	35.5
Dynamic A	80.4	70.1	74.2	**35.6**
Dynamic B	**85.0**	68.7	73.8	35.4
Dynamic C	82.4	**70.6**	74.7	35.0

Table 4.1: Evaluation of Precision, Recall and AP for the POG dataset and its altitude-dependant subsets with different dynamic loss gains.

4.3.3 Altitude Injection

From the results in Table 4.2 we can see that the performance measured by AP_{50} and AP improved compared to the baseline when we concatenated the

altitude information with the image at the input layer. This was also true for the low and high altitude levels when tested on the altitude-specific subsets. Injecting the altitude information at the neck or head as described in Section 3.5.3 resulted in a worse AP_{50} performance, although injecting it at the head yielded a higher AP than the baseline. The changes in FLOPS and parameters were only marginal when adapting the architecture for altitude injection so they were not taken into consideration when evaluating the performance of this method.

Full	*P*	*R*	AP_{50}	*AP*
Baseline	89.5	**78.2**	84.5	42.5
Input	**92.7**	76.2	**84.8**	**43.1**
Neck	90.9	77.7	84.3	41.7
Head	90.3	77.3	83.8	42.9

Medium	*P*	*R*	AP_{50}	*AP*
Baseline	90.7	83.0	**88.7**	43.2
Input	88.5	**84.9**	87.9	43.4
Neck	**91.1**	83.7	88.1	42.1
Head	90.0	83.3	87.6	**43.6**

Low	*P*	*R*	AP_{50}	*AP*
Baseline	**97.6**	87.2	93.5	54.0
Input	97.3	**92.0**	**95.4**	**55.3**
Neck	97.0	90.5	94.1	53.2
Head	94.5	89.9	94.0	54.4

High	*P*	*R*	AP_{50}	*AP*
Baseline	84.6	68.4	73.9	34.7
Input	79.5	**70.9**	**74.1**	**34.9**
Neck	83.3	68.7	73.2	33.4
Head	**85.3**	65.7	72.7	**34.9**

Table 4.2: Evaluation of Precision, Recall and AP for the POG dataset and its altitude-specific subsets with the altitude injected at the different layers.

In Table 4.3 we can observe a similar result for the comparison of the baseline model with the altitude-aware model on the HIT-UAV dataset.

	mAP_{50}	mAP
Baseline	81.1	50.8
Input	**85.3**	**54.9**

Table 4.3: Performance of YOLOv7-tiny and the model that included the altitude information at the input layer on the full HIT-UAV dataset.

4.3.4 Altitude Prediction

With an altitude loss gain of $\gamma_{alt} = 0.01$, for *Auxiliary A*, the model with the auxiliary altitude prediction heads was able to predict the altitude of an image correctly with an accuracy of 74%. These altitude predictions are visualized in Figure 4.4. From the confusion matrix, we can further observe that the wrong predictions at the highest and lowest altitudes are significantly lower for the opposite extreme than for the intermediate levels. In other words, when the image is labeled as low altitude, the model is less likely to classify it as

Full	*P*	*R*	AP_{50}	*AP*
Baseline A	89.5	78.2	84.5	42.5
Auxiliary A	89.7	**79.0**	**85.2**	**42.6**
Auxiliary B	**90.1**	75.6	82.5	41.2

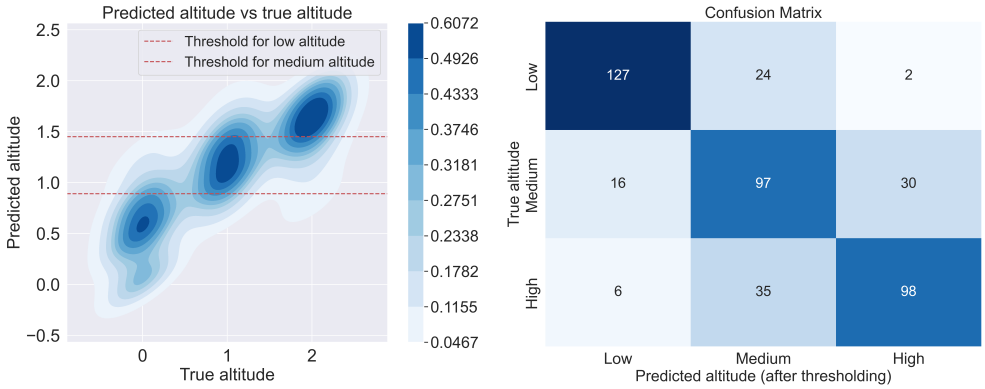
Medium	*P*	*R*	AP_{50}	*AP*
Baseline A	90.7	83.0	88.7	43.2
Auxiliary A	**93.5**	83.7	**89.4**	**43.6**
Auxiliary B	87.9	**84.1**	88.1	42.3

Low	*P*	*R*	AP_{50}	*AP*
Baseline A	**97.6**	87.2	93.5	54.0
Auxiliary A	95.6	**90.7**	**94.2**	**54.5**
Auxiliary B	97.2	88.9	93.4	52.8

High	*P*	*R*	AP_{50}	*AP*
Baseline A	**84.6**	68.4	73.9	**34.7**
Auxiliary A	78.0	**72.8**	**74.2**	33.4
Auxiliary B	80.9	64.5	69.2	32.5

Table 4.4: Comparison of the object detection results for the model with an auxiliary head to predict the altitude.

high altitude compared to medium altitude and vice-versa. For *Auxiliary B* with a gain of $\gamma_{alt} = 0.02$, we observed a decrease in AP_{50} and AP for all altitude levels. Even though more weight was attributed to the altitude loss, the accuracy of the altitude prediction also suffered and resulted in only 63% of the altitudes being correctly predicted. The adaptations in the architecture for the altitude prediction task resulted in an increase of around 5% in the number of parameters and FLOPS.



(a) (b)

Figure 4.4: Density plot (a) and confusion matrix (b) for altitude prediction of model with the auxiliary head and $\gamma_{alt} = 0.01$ on the POG dataset. The dashed lines indicate where we set the threshold to categorize the average of the predicted values to a discrete altitude level.

4.3.5 Combination of Altitude Injection and Prediction

The results of combining the two previous methods of Altitude Injection 3.5.3 and Altitude Prediction 3.5.4 can be seen in Table 4.5. We can observe that the baseline model performed better in AP_{50} and AP in all instances compared

to the three proposed methods. When only comparing the proposed methods with each other the model that had the altitude information injected at the head performed best, while the model with the altitude information given at the input layer performed the worst.
Regarding the accuracy of the altitude prediction task for the auxiliary head, our findings showed an accuracy of 100%. This means the model could identify the altitude level for all images in the test set correctly. In Figure 4.5 we can see this being visualized through the loss function. Compared to the altitude validation loss when not using the altitude as an additional input, after initially staying at a similar loss value, the training and validation loss of the combinatory approaches sharply decrease towards 0 after several epochs.

Full	*P*	*R*	AP_{50}	*AP*
Baseline	89.5	78.2	**84.5**	**42.5**
Input	88.7	74.9	80.5	39.5
Neck	**91.4**	77.0	83.4	41.1
Head	88.6	**79.2**	84.1	41.8

Medium	*P*	*R*	AP_{50}	*AP*
Baseline	90.7	**83.0**	**88.7**	**43.2**
Input	92.3	78.9	85.4	41.2
Neck	**93.9**	81.4	87.1	42.0
Head	92.4	81.7	87.9	42.1

Low	*P*	*R*	AP_{50}	*AP*
Baseline	**97.6**	87.2	93.5	**54.0**
Input	94.0	89.3	92.7	51.9
Neck	94.7	90.1	93.1	53.1
Head	94.5	**90.3**	**93.6**	53.4

High	*P*	*R*	AP_{50}	*AP*
Baseline	**84.6**	68.4	**73.9**	**34.7**
Input	82.1	61.3	67.0	29.5
Neck	78.9	70.1	73.0	31.9
Head	80.3	**71.0**	73.6	34.2

Table 4.5: Comparison of the object detection results for the model combining the auxiliary head and $\gamma_{alt} = 0.01$ with the altitude information injected at the layer referenced by the names in the Table. The performance is tested on the full and altitude specific subsets of the POG dataset.

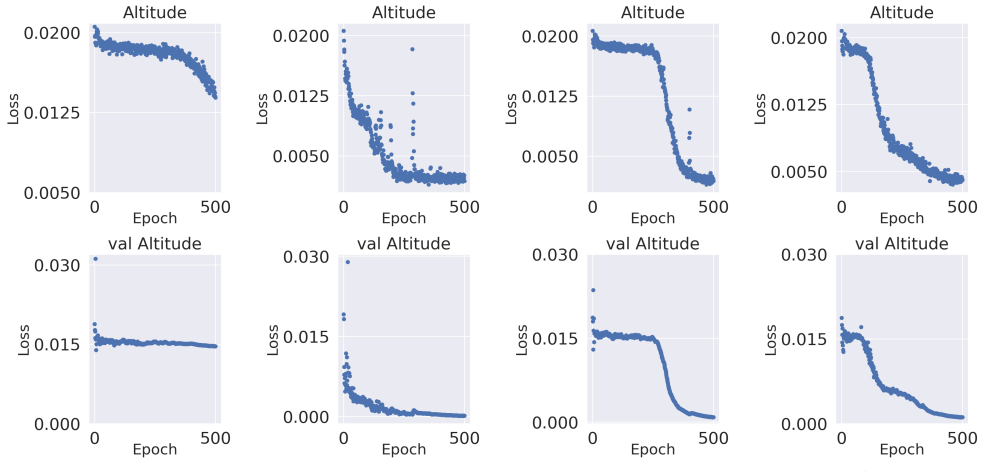


Figure 4.5: Comparison of altitude training and validation loss with a gain $\gamma_{alt} = 0.01$ between the implicit learning approach where the model only learns it through the auxiliary head (a) and the explicit *i.e.,* combinatory approaches where the altitude information is also injected into the network at the input (b), neck (c), and head (d).

4.4 Exploratory Analysis

This Section reports the results of the exploratory analysis described in 3.6 and gives insights into the performance of some non-altitude-adaptive but still related methods.

4.4.1 Removing the FPN layers

When the FPN layers of the YOLOv7-tiny network architecture are removed we can observe the results detailed in 4.6. The AP_{50} metric describing the object detection performance of the baseline is better than that of the model with the neck layers removed. This drop in AP_{50} increases with altitude. On the other hand, the number of parameters and FLOPS are significantly reduced due to the network being smaller.

Model	#Params.	GFLOPS	AP_{50}	AP^L_{50}	AP^M_{50}	AP^H_{50}
Base	6.01M	13.1	**84.5**	**93.5**	**88.7**	**73.9**
No-FPN	**5.37M**	**12.0**	82.2	91.9	86.4	71.2
Change	↑ 10.65%	↑ 8.40%	↓ 2.72%	↓ 1.71%	↓ 2.59%	↓ 3.65%

Table 4.6: Evaluation on model complexity metrics and average precision on POG as well as its altitude-specific subsets of the model with the FPN removed compared to the base model.

4.4.2 Multi-experts

The expert models were fine-tuned on their respective altitude-specific subset of the full dataset and trained for 200 epochs from the best 300 epochs baseline model to reach a total of 500 trained epochs just like the baseline model. The backbone layers, more specifically the first 11 layers when referring to Figure 2.1, of the pre-trained YOLOv7-tiny model were frozen when we fine-tuned the expert models. Looking at Table 4.7 it can be seen that the highest results for each column are on the main diagonal which means each expert model performs best on their specific task. However, they perform significantly worse on all other tasks, with the lowest results at the furthest altitude from their task, *i.e.,* the expert on low altitude performs worst at high altitudes and vice versa.

Model	Full	Low	Medium	High
Baseline	**84.5**	93.5	88.7	73.9
Expert-low	65.6	**94.2**	73.9	36.3
Expert-medium	78.3	84.9	**89.4**	62.5
Expert-high	70.3	54.4	75.4	**74.1**

Table 4.7: Comparison of AP_{50} results of the models that were fine-tuned on an altitude-specific subset of POG.

They share around 52% of the total number of parameters. So in practice, if the backbone is shared and then the following layers are chosen according to the altitude this would result in a parameter increase of 96% while the general performance increase is estimated to be less than 2%.

4.4.3 Additional Head

For the final experiment we investigated the PDWT-YOLO [29] approach using their official implementation*. From Table 4.8 we can observe that while PDWT-YOLO did perform significantly better on the POG dataset it performed significantly worse on the HIT-UAV and FLOX dataset. On the other hand, with more than four times fewer parameters, our P2-YOLO outperforms both PDWT-YOLO and the baseline on all the datasets.

			mAP_{50}			
Model	#Params.	GFLOPS	UAVDT	POG	HIT	FLOX
YOLOv7-tiny	6.01M	13.1	99.3	84.5	81.1	86.6
PDWT-YOLO	6.14M	24.2	98.1	92.0	72.5	70.4
P2-YOLO	**1.52M**	**11.8**	**99.5**	**94.9**	**83.9**	**88.3**

Table 4.8: Evaluation of mAP_{50} results of YOLOv7-tiny, PDWT-YOLO and P2-YOLO on the different datasets.

4.5 Final Evaluation

If we were to compare all the investigated methods and order them according to their performance then we would get the results shown in Table 4.9. Note that we did not include the dynamic loss method in this comparison because

*https://github.com/1thinker1/PDWT-YOLO (accessed on 12 June 2023)

those trained models had different hyperparameters and we did not include the frozen layer approach because we did not have a system in place that would dynamically load the appropriate fine-tuned weights for each altitude level and thus give us the performance on the full dataset. Furthermore, when comparing the altitude-adaptive methods we would like to point out that although the altitude prediction method has a higher AP_{50} value, the altitude injection approach has a higher AP value.

Model	AP_{50}	AP
Dynamic anchors	60.8	18.6
No-FPN	82.2	38.4
Combination	83.4	41.1
Baseline	84.5	42.5
Injection@input	84.8	43.1
Prediction@0.01	85.2	42.6
PDWT-YOLO	92.0	44.4
P2-YOLO	**94.9**	**52.5**

Table 4.9: Comparison of the best models on POG for the different methods discussed in this thesis with the exception of the dynamic loss model and the frozen layer approach.

Chapter 5

Discussion

The upcoming sections aim to explain and delve into the implications of the results presented in the previous Chapter. We follow the same structure as in Chapter 4 discussing the findings of each separate method before summarizing the key insights and takeaways from all the experiments.

5.1 Altitude-Adaptive Analysis

The results of the altitude-adaptive analysis presented in Section 4.3 were varied. While in some cases a proposed method improved the performance compared to the baseline, in multiple other experiments the baseline could not be improved with the proposed changes and even deteriorated. Nevertheless, these results still provide valuable insights into which and how much the evaluation metrics changed or how the model behaved during training. Furthermore, by reporting the results of the models on the different altitude levels as well as the full dataset we can better discuss where the positive or negative impact of the proposed methods is greatest.

5.1.1 Dynamic Anchors

The results in 4.3.1 highlighted that in the way we implemented it, the dynamic anchors are not a promising approach. While we were able to validate that adjusting the anchor size would make the model perform better at certain altitudes this is accompanied by a trade-off in general detection performance which would make a general set of anchors better suited than one more altitude-specific set. The idea to circumvent that problem through multiple sets of anchors that get dynamically chosen depending on the altitude did not

bear positive results. This is likely because the values of the weights of the model are optimal for all sets of anchors, which results in them not being optimal for each set of anchors individually, and thus instead of increasing performance, it inhibits object detection. This is highlighted by the fact that the validation losses converged to a value that was significantly higher than those of the baseline.

While it is true that a good choice of anchors is imperative to train an object detection model, the number of anchors also plays a role, and increasing that number can eventually decrease performance [21]. While the dynamic approach does not change the number of anchors for a given input, the total number of anchors and thus the number of anchors the model has to optimize for in the training process does increase. Thus, after evaluating the results of our experiments, when it concerns anchors, a general set of anchors that is optimized for the whole dataset might be better suited than multiple sets of anchors that are only optimized for specific subsets.

5.1.2 Dynamic Loss

The findings of the experiments using a dynamic loss approach gave some informative insights into how the objectness loss influences precision, recall, and average precision for different capture altitudes of the images.

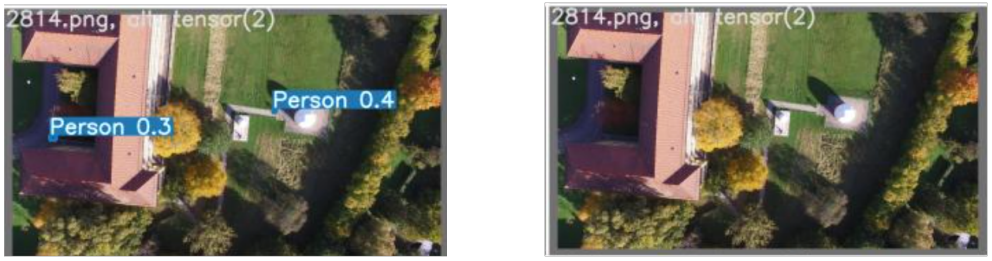
Contrary to our initial intuition an overall increase in objectness loss gain ($\gamma_{obj}(1)$ to $\gamma_{obj}(1.5)$) did not improve recall. While a higher gain should penalize the model more when it fails to predict the presence of an object, the results instead show a decrease in recall with that higher gain. While the recall does increase with higher gain for low altitude images, which might be the most similar ones to ground images, at medium and high altitudes, the recall decreased. This decrease at higher altitudes could be explained by the assumption, that at higher altitudes the threshold of objectness becomes too high causing the model to miss detections by not confidently predicting the objectness score. Since the localization accuracy might suffer due to an increased focus on objectness, this could also cause the model to not correctly localize an object, leading to missed detections and an alternate explanation of the lower recall. However, while the experiments have shown that the correlation between recall, precision, and AP is more intricate than initially assumed, we could still show a relation between the different metrics for different altitude levels and varying objectness loss gain values.

Another interesting finding of this experiment was that a model that had quantitatively the highest recall at high altitudes did not always correlate to

fewer objects being missed. When performing a qualitative analysis of the network's prediction it could be observed that the lower recall model detected objects that multiple models struggled to detect, while the higher recall model missed them. An example of this can be seen in Figure 5.1.

While we were not able to improve the baseline performance with this method, the experiments still gave valuable insights into how the objectness loss influences precision, recall, and AP at different altitudes and how varying the gain dynamically influences these factors as well. This knowledge can potentially be used to further optimize a model for specific use cases through an additional hyperparameter.

Finally, it should also be noted that in our implementation the dynamic loss could only be varied per batch which did not always only include images of the same altitude level. Thus, the results might be improved if the loss gain could be adapted for each image instead to most accurately execute this approach. Furthermore, for the best results, it would be wise to first do a hyperparameter search to get the best objectness loss gain in general and then get the best objectness loss gain for the different altitude levels to get a final evaluation on whether a model with an optimized dynamic loss could outperform an optimized baseline model.



(a) (b)

Figure 5.1: Comparison of detection results of highest and lowest reported recall for high altitude level from the experiments reported in 4.3.2. The baseline model with $\gamma_{obj}(1.5)$ correctly detects the two object instances although it has a lower recall at this altitude level than the dynamic model with $\gamma_{dyn}(0.67, 1.5)$ which misses both instances although it has the highest recorded recall.

5.1.3 Altitude Injection

The results for the altitude injection approach showed, that concatenating the altitude information with the image before it is fed through the first layer of the network can improve the performance. While the greatest

performance increase could be observed for the low-altitude images, the additional information also improved performance at high altitudes indicating that feeding the altitude information to the model helps it learn features that improve its detection performance. At medium altitudes, however, the performance of that model is worse than the baseline. The reason for that might lie in the fact that medium-altitude images have some overlap with low and even more so with high-altitude images as seen in Figure 4.1.

By training a baseline model and an altitude-injection model on a different dataset and observing a similar performance increase we are able to further support the findings that our proposed method can improve object detection. Observing an increase in detection performance on more than one dataset suggests that this is an approach that could likely be transferred to other UAV-based datasets which include altitude information.

The experiment also generated insights into how the altitude injection point influences the resulting performance. When injecting the altitude information at either the neck or the head, the object detection performance of the model decreased. Observing that the injection at the head performed the worst, it can be assumed, that the later the altitude information was given to the network, the less it was able to extract helpful features from it.

5.1.4 Altitude Prediction

The results presented in Section 4.3.4 reveal insights into whether or not the YOLOv7-tiny model was able to predict the altitude from the image alone. The findings showed that while not excelling at predicting the altitude from the image alone, the model was still able to predict the altitude to some extent. With three altitude levels, the probability of predicting the correct altitude level by random chance or constantly predicting one altitude level would be around 33%. So with an accuracy of more than 70%, we can assume that the model learned some features from the input that let it determine the correct altitude.

While the network did not always predict the correct altitude values, this additional task has helped some of the models improve their object detection performance. This demonstrates that the model does not necessarily need to receive the altitude information as input to become altitude-aware. Additionally, it shows that training an object detector on an additional altitude label likely makes the model form a meaningful connection between the objects in an image and the altitude it was captured. The analysis further details that the gain for the altitude loss is a relevant parameter that influences the altitude prediction accuracy as well as the detection performance and is

thus an additional hyperparameter that requires careful tuning. We can note that, since the altitude prediction is only a means to eventually improve the object detection performance, the loss gain should be chosen in such a way that the altitude loss will not outweigh the other three losses.

5.1.5 Combination of Altitude Injection and Prediction

Given the results in Section 4.3.5 and an accuracy of 100% for the altitude prediction task we can safely assume that the model learned the correlation between the additional altitude input and predicting that value through the auxiliary head at the output layer. However, all the tested models performed worse than the baseline, which suggests that the model learned to pass that information along without learning additional features that are relevant to object detection. Figure 4.5 shows that depending on where the additional input was injected, the model learns the correlation between it and its prediction faster or slower. Contrary to how fast or slow the model seems to learn this correlation, the earlier the additional input is given the worse the observed detection performance was. This further indicates that the model learned to simply pass the information along without gaining additional insights, meaning the earlier the input, the more weights are required to store this information inhibiting the network's ability to learn object detection-related features. In summary, while the earlier experiments showed that giving the network altitude information as input or using it as labels to train an auxiliary head can improve model performance, combining both approaches more likely disturbs the learning process instead of benefiting it.

5.2 Exploratory Analysis

The findings of Section 4.4 helped us evaluate and benchmark some related methods that do not directly incorporate altitude into the training process. Using the same experimental setup as the previous experiments, helped us set into relation the performance of alternative methods to our proposed altitude-adaptive methods. Through the results, we were able to understand the impact of the FPN layers and fine-tuning on different altitude levels as well as how well PDWT-YOLO works across different UAV datasets.

5.2.1 Removing the FPN layers

The results of removing the FPN layers were straightforward. The improvements in the number of parameters and floating point operations were simply a natural consequence of removing layers from the network architecture. While a performance decrease was anticipated at lower altitudes because there is more likely to be a size variance due to the camera angle being shallower but the performance decreased similarly or even more at higher altitudes. So the reason why the performance for all altitude levels decreased could mean that the FPN adds contextual information to the features that are relevant even at higher altitudes where there is less scale variation. Nevertheless, this experiment has given insights into the trade-off between performance and network size and speed for a UAV-based dataset. This could help adapt the middle layers of the YOLOv7-tiny architecture to be more effective in increasing performance with the same increase in parameters.

5.2.2 Multi-experts

The results presented in 4.4.2 showed that using a fine-tuned model for each altitude level improves the performance for that level. However, this performance increase is traded off by a significant parameter increase for an overall system that grows linearly with the number of altitude levels. While the parameters at run-time do not increase, the additionally trained weights have to be stored on the device. With memory being limited on edge devices this trade-off could make it challenging to store all the parameters on the device. Even if there was enough space, with this level of performance increase compared to the additional space needed, it might be more sensible to use a more complex architecture for a greater performance increase. Although the advantage this multi-expert approach hold over a more complex architecture is that the inference speed stays the same as the base model because the number of parameters at run-time does not change.

5.2.3 Additional Head

While the theory and reasoning behind the PDWT-YOLO network have been detailed by Zhang *et al.* [29] the findings in our work have only been able to validate the improvements for some datasets. From our experiments, it seems that not all UAV-based datasets can benefit from their proposed method as the HIT-UAV and the FLOX datasets saw a significant drop in performance. The reason for this performance drop is unclear, however, it is noteworthy to

mention that the most significant decrease was observed in the two datasets that consisted of thermal images. On the other hand, the rather simple architectural change of adding a small object detection head and removing the largest object detection head as described in Section 3.6.3 inspired by the PDWT-YOLO and the other models mentioned in Section 2.4 as related work, significantly outperformed both the baseline model and the PDWT-YOLO on all the tested datasets. Furthermore, this improvement was not only displayed through a performance increase but also a significant reduction in the number of parameters. This further supports the findings that the shallower layers can be more relevant in UAV-based object detection than the deeper layers. Adding on top that the improvement of this simpler approach were greater than the more complex PDWT-YOLO highlights the importance of interpretable and modular model architectures for object detection depending on the specific task and dataset.

5.3 Key insights and takeaways

- While anchors play a crucial role in object detection, a more general and smaller set of anchors beats a larger set of more specific anchors. This also includes multiple equally sized but more specific anchor sets that are changed dynamically.
- A dynamic objectness loss gain influences detection performance for the different altitude levels. On a full dataset across all altitudes, a constant objectness loss performed better or equivalent to the different dynamic losses. While the dynamic loss can give better control of precision and recall for different altitude levels, this does not guarantee more objects being detected at these altitudes.
- Concatenating the altitude with the input can improve the general detection performance of the network while injecting the information at a different layer in the network worsens it.
- YOLOv7-tiny with an auxiliary altitude head can, to a certain degree, learn to predict the altitude level from the image alone and this can improve the object detection performance when the altitude loss gain is tuned correctly.
- Although the altitude prediction method has a higher increase in the AP_{50} metric, the altitude injection method has a greater increase in AP

while having a similar number of parameters and FLOPS compared to the baseline.

- When injecting the model with the altitude as well as using an auxiliary head to predict the altitude, the model learns to propagate the altitude information through the network. However, this results in a performance drop for the object detection task
- The FPN network inside YOLOv7-tiny is still relevant even for UAV-based images.
- Sharing the backbone layers and then fine-tuning on altitude-specific subset improves performance for that specific subset. However, the performance increase might be offset by the overall increase in parameters in practice.
- Network changes that address some of the challenges of UAV-based images by their root causes have shown significantly more positive results on the datasets. Although these changes can cause major improvements they can be also more detrimental and thus not universally applicable for all UAV datasets such as the case for the PDWT-YOLO on the HIT-UAV and FLOX datasets.
- However, architectural changes such as adding and removing particular detection heads even without further modifications can reduce the complexity of the model while significantly improving the detection performance on a variety of UAV datasets.
- Overall the best model tested in this thesis was the P2-YOLO, although this approach is not mutually exclusive with the altitude-adaptive methods

Chapter 6

Conclusions and Future work

This chapter concludes the thesis by first presenting the general conclusions in Section 6.1. Following this, the limitations of this thesis will be addressed in Section 6.2, highlighting the constraints and suggesting areas for further investigation. Finally, Section 6.3 explores potential future research to expand upon this study and delve deeper into the subject matter.

6.1 Conclusion

The thesis work answered the research question presented in Section 1.2 on whether or not it is possible to make use of altitude information to improve detection performance through dynamic or learning-based approaches. The investigation involved implementing several ways to make use of this altitude information and analyzing the results. The extensive experiments gave insights into more and less promising approaches and how they influence performance.

We found out that instead of being able to make optimal use of specific anchors for specific situations the model converged to a suboptimal state with our dynamic anchor approach that was significantly worse than a single more general anchor set. The dynamic loss approach could be promising for separately fine-tuning precision or recall at different altitude levels but led to an overall decrease in performance in our experiments. On the other, our research has shown that giving an object detection model altitude information as additional input or using it to train an auxiliary head improves the detection performance of the trained model in some cases. A combination of both approaches, however, produces a worse object detector.

We also confirmed that using the YOLOv7-tiny architecture and fine-tuning separate models for specific altitudes while using a shared backbone

can slightly improve capabilities. However, this leads to a larger and more complicated overall object detection system as a noteworthy trade-off. Additionally, our research revealed that the FPN layers of the network are still relevant even for UAV-based object detection, and removing them comes with a reduction in mean average precision that is greater at higher altitude levels. Finally, the experiments we performed showed that adding an extra small object detection head in place of the largest detection head produced the most significant improvements for UAV-based object detection, while additional proposed changes as in the case of the PDWT-YOLO can be detrimental to the model's performance on some aerial datasets.

6.2 Limitations

This study was subject to several limitations which were primarily based on resources and time. First and foremost, the analysis documented in this thesis focuses mainly on the POG dataset. To make sure a model is well-trained for a fair comparison it should be trained until convergence and on sufficient data to decrease the standard deviation which is time and resource-intensive since we only assumed the standard deviation for the other approaches would be similar to the baseline but did not test it. To ensure comparability across the different methods, we chose to use only one of the initially discussed datasets and limited the use of the other datasets for smaller preliminary studies. While this approach provides valuable insights, broadening the scope of experiments would ensure a more comprehensive understanding of our findings. Additionally, while the standard deviation of the model's results should not vary to influence the findings when training long enough, it nevertheless limits the validity to a certain degree. Furthermore, each method has multiple ways of approaching them of which we chose a subset that we believed most promising. This means that even when the results turned out to be negative, it does not fully rule out the possibility of a similar method working under different conditions or with alternative configurations. Finally, while this study focused on exploring multiple different methods for the chosen model, more comprehensive insights could be achieved by testing multiple models. Evaluating different models would allow us to understand the generalizability of our findings across different architectures.

6.3 Future Work

Building on the findings of this thesis, future work could address the limits outlined earlier by expanding the scope of the study to include more datasets and models. To improve the reliability of the findings multiple reruns of the experiments could be conducted to get a better estimate of mean and standard deviation of the resulting metrics.

Shortcomings of the experimental setups could also be addressed by refining the implementations such as adapting the dynamic loss for each image instead of using the average over the batch with a threshold. In addition, in our altitude-dependant methods we only considered three discrete altitude levels but with more precise information available, future work could also include analyzing how finer or even continuous levels of altitude influence the performance of the proposed methods. Furthermore, the integration of contextual information in the learning process of the network does not have to be limited to the altitude but could also include information about the viewing angle, weather, time of day and more.

Finally, subsequent research could investigate combining the most promising methods found in this work such as the P2-YOLO together with the altitude prediction or injection approach.

References

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, ISSN: 1063-6919, Jun. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848. [Online]. Available: https://ieeexplore.ieee.org/document/5206848 (visited on 07/07/2024).

[2] T.-Y. Lin *et al.*, "Microsoft COCO: Common Objects in Context," en, in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., Cham: Springer International Publishing, 2014, pp. 740–755, ISBN: 978-3-319-10602-1. DOI: 10.1007/978-3-319-10602-1_48. [Online]. Available: https://doi.org/10.1007/978-3-319-10602-1_48.

[3] H.-Y. Hou *et al.*, "Ensemble Fusion for Small Object Detection," in *2023 18th International Conference on Machine Vision and Applications (MVA)*, Jul. 2023, pp. 1–6. DOI: 10.23919/MVA57639.2023.10215748. [Online]. Available: https://ieeexplore.ieee.org/document/10215748 (visited on 01/23/2024).

[4] A. M. Rekavandi, S. Rashidi, F. Boussaid, S. Hoefs, E. Akbas, and M. bennamoun, *Transformers in Small Object Detection: A Benchmark and Survey of State-of-the-Art*, arXiv:2309.04902 [cs], Sep. 2023. DOI: 10.48550/arXiv.2309.04902. [Online]. Available: http://arxiv.org/abs/2309.04902 (visited on 01/23/2024).

[5] D. Cazzato, C. Cimarelli, J. L. Sanchez-Lopez, H. Voos, and M. Leo, "A Survey of Computer Vision Methods for 2D Object Detection from Unmanned Aerial Vehicles," en, *Journal of Imaging*, vol. 6, no. 8, p. 78, Aug. 2020, Number: 8 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 2313-433X. DOI: 10.3390/jimaging6080078. [Online]. Available: https://www.mdpi.com/2313-433X/6/8/78 (visited on 06/04/2024).

[6] V. Zaliva and F. Franchetti, "Barometric and GPS altitude sensor fusion," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, ISSN: 2379-190X, May 2014, pp. 7525–7529. DOI: 10.1109/ICASSP.2014.6855063. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6855063 (visited on 07/17/2024).

[7] T. Matyja, Z. Stanik, and A. Kubik, "Automatic correction of barometric altimeters using additional air temperature and humidity measurements," en, *GPS Solutions*, vol. 28, no. 1, p. 40, Dec. 2023, ISSN: 1521-1886. DOI: 10.1007/s10291-023-01582-7. [Online]. Available: https://doi.org/10.1007/s10291-023-01582-7 (visited on 07/14/2024).

[8] A. G. Dempster and E. Cetin, "Interference Localization for Satellite Navigation Systems," *Proceedings of the IEEE*, vol. 104, no. 6, pp. 1318–1326, Jun. 2016, ISSN: 1558-2256. DOI: 10.1109/JPROC.2016.2530814. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7439734 (visited on 07/14/2024).

[9] P. Zhu *et al.*, "VisDrone-DET2018: The Vision Meets Drone Object Detection in Image Challenge Results," in *Computer Vision - ECCV 2018 Workshops - Munich, Germany, September 8-14, 2018, Proceedings, Part V*, L. Leal-Taixé and S. Roth, Eds., ser. Lecture Notes in Computer Science, vol. 11133, Springer, 2018, pp. 437–468. DOI: 10.1007/978-3-030-11021-5_27. [Online]. Available: https://doi.org/10.1007/978-3-030-11021-5%5C_27.

[10] L. A. Varga, B. Kiefer, M. Messmer, and A. Zell, "SeaDronesSee: A Maritime Benchmark for Detecting Humans in Open Water," in *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, Jan. 2022, pp. 3686–3696, ISBN: 2642-9381. DOI: 10.1109/WACV51458.2022.00374.

[11] M. Hu, Z. Li, J. Yu, X. Wan, H. Tan, and Z. Lin, "Efficient-Lightweight YOLO: Improving Small Object Detection in YOLO for Aerial Images," *Sensors (Basel, Switzerland)*, vol. 23, no. 14, p. 6423, Jul. 2023, ISSN: 1424-8220. DOI: 10.3390/s23146423. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10385816/ (visited on 02/16/2024).

[12] Z. Wu, K. Suresh, P. Narayanan, H. Xu, H. Kwon, and Z. Wang, "Delving Into Robust Object Detection From Unmanned Aerial Vehicles: A Deep Nuisance Disentanglement Approach," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Nov. 2019, pp. 1201–1210. DOI: 10.1109/ICCV.2019.00129. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/ICCV.2019.00129.

[13] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature Pyramid Networks for Object Detection," en, in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI: IEEE, Jul. 2017, pp. 936–944, ISBN: 978-1-5386-0457-1. DOI: 10.1109/CVPR.2017.106. [Online]. Available: http://ieeexplore.ieee.org/document/8099589/ (visited on 06/06/2024).

[14] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017, ISSN: 1939-3539. DOI: 10.1109/TPAMI.2016.2577031. [Online]. Available: https://ieeexplore.ieee.org/document/7485869 (visited on 06/06/2024).

[15] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, ISSN: 1063-6919, Jun. 2016, pp. 779–788. DOI: 10.1109/CVPR.2016.91. [Online]. Available: https://ieeexplore.ieee.org/document/7780460 (visited on 09/17/2024).

[16] W. Liu *et al.*, "SSD: Single Shot MultiBox Detector," en, in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds., Cham: Springer International Publishing, Sep. 2016, pp. 21–37, ISBN: 978-3-319-46448-0. DOI: 10.1007/978-3-319-46448-0_2.

[17] Z. Liu *et al.*, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2021, pp. 9992–10 002, ISBN: 2380-7504. DOI: 10.1109/ICCV48922.2021.00986.

[18] C. -Y. Wang, A. Bochkovskiy, and H. -Y. M. Liao, "YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2023, pp. 7464–7475, ISBN: 2575-7075. DOI: 10.1109/CVPR52729.2023.00721.

[19] M. Gao, Y. Du, Y. Yang, and J. Zhang, "Adaptive anchor box mechanism to improve the accuracy in the object detection system," *Multimedia Tools and Applications*, vol. 78, Oct. 2019. DOI: 10.1007/s11042-019-07858-w.

[20] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren, "Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, Feb. 2020, pp. 12 993–13 000. DOI: 10.1609/aaai.v34i07.6999.

[21] T. Yang, X. Zhang, W. Zhang, and J. Sun, "MetaAnchor: Learning to Detect Objects with Customized Anchors," in *Neural Information Processing Systems*, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:49564144.

[22] X. Zhang, F. Wan, C. Liu, X. Ji, and Q. Ye, "Learning to Match Anchors for Visual Object Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 6, pp. 3096–3109, Jun. 2022, ISSN: 1939-3539. DOI: 10.1109/TPAMI.2021.3050494.

[23] R. Jin and D. Lin, "Adaptive Anchor for Fast Object Detection in Aerial Image," *IEEE Geoscience and Remote Sensing Letters*, vol. 17, no. 5, pp. 839–843, May 2020, ISSN: 1558-0571. DOI: 10.1109/LGRS.2019.2936173. [Online]. Available: https://ieeexplore.ieee.org/document/8824218 (visited on 03/26/2024).

[24] Y. Zhao, H. Su, C. Zhong, W. Zou, Y. Guo, and S. Liu, *Iac: Iou-Aware Clustering for Anchor Configuration Determination in Efficient Defect Detection*, en, 2024. DOI: 10.2139/ssrn.4785165. [Online]. Available: https://www.ssrn.com/abstract=4785165 (visited on 04/16/2024).

[25] X. Fu, G. Wei, X. Yuan, Y. Liang, and Y. Bo, "Efficient YOLOv7-Drone: An Enhanced Object Detection Approach for Drone Aerial Imagery," en, *Drones*, vol. 7, no. 10, p. 616, Oct. 2023, Number: 10 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 2504-446X. DOI: 10.3390/drones7100616. [Online]. Available: https://www.mdpi.com/2504-446X/7/10/616 (visited on 04/16/2024).

[26] Z. Wang, Z. Liu, G. Xu, and S. Cheng, "Object Detection in UAV Aerial Images Based on Improved YOLOv7-tiny," in *2023 4th International Conference on Computer Vision, Image and Deep Learning (CVIDL)*, May 2023, pp. 370–374. DOI: 10.1109/CVIDL58838.2023.10166362. [Online]. Available: https://ieeexplore.ieee.org/document/10166362 (visited on 03/26/2024).

[27] Z. Tong, Y. Chen, Z. Xu, and R. Yu, *Wise-IoU: Bounding Box Regression Loss with Dynamic Focusing Mechanism*, arXiv:2301.10051 [cs] version: 3, Apr. 2023. DOI: 10.48550/arXiv.2301.10051. [Online]. Available: http://arxiv.org/abs/2301.10051 (visited on 09/22/2024).

[28] Y. Wang, H. Zou, M. Yin, and X. Zhang, "SMFF-YOLO: A Scale-Adaptive YOLO Algorithm with Multi-Level Feature Fusion for Object Detection in UAV Scenes," en, *Remote Sensing*, vol. 15, no. 18, p. 4580, Jan. 2023, Number: 18 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 2072-4292. DOI: 10.3390/rs15184580. [Online]. Available: https://www.mdpi.com/2072-4292/15/18/4580 (visited on 02/01/2024).

[29] L. Zhang, N. Xiong, X. Pan, X. Yue, P. Wu, and C. Guo, "Improved Object Detection Method Utilizing YOLOv7-Tiny for Unmanned Aerial Vehicle Photographic Imagery," en, *Algorithms*, vol. 16, no. 11, p. 520, Nov. 2023, Number: 11 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 1999-4893. DOI: 10.3390/a16110520. [Online]. Available: https://www.mdpi.com/1999-4893/16/11/520 (visited on 04/17/2024).

[30] Z. Zhang, "Drone-YOLO: An Efficient Neural Network Method for Target Detection in Drone Images," en, *Drones*, vol. 7, no. 8, p. 526, Aug. 2023, Number: 8 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 2504-446X. DOI: 10.3390/drones7080526. [Online]. Available: https://www.mdpi.com/2504-446X/7/8/526 (visited on 02/16/2024).

[31] Z. Gevorgyan, "SIoU Loss: More Powerful Learning for Bounding Box Regression," *ArXiv*, vol. abs/2205.12740, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:249063031.

[32] I. Martinez-Alpiste, G. Golcarenarenji, Q. Wang, and J. M. Alcaraz Calero, "Altitude-Adaptive and Cost-Effective Object Recognition in an Integrated Smartphone and UAV System," in *2020 European Conference on Networks and Communications (EuCNC)*, ISSN: 2575-4912, Jun. 2020, pp. 316–320. DOI: 10.1109/EuCNC48522.2020.9200951. [Online]. Available: https://ieeexplore.ieee.org/document/9200951 (visited on 03/21/2024).

[33] W. He, Z. Huang, Z. Wei, C. Li, and B. Guo, "TF-YOLO: An Improved Incremental Network for Real-Time Object Detection," en, *Applied Sciences*, vol. 9, no. 16, p. 3225, Jan. 2019, Number: 16 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 2076-3417. DOI: 10.3390/app9163225. [Online]. Available: https://www.mdpi.com/2076-3417/9/16/3225 (visited on 07/02/2024).

[34] B. Kiefer, M. Messmer, and A. Zell, "Diminishing Domain Bias by Leveraging Domain Labels in Object Detection on UAVs," in *2021 20th International Conference on Advanced Robotics (ICAR)*, Dec. 2021, pp. 523–530. DOI: 10.1109/ICAR53236.2021.9659357.

[35] M. Messmer, B. Kiefer, and A. Zell, "Gaining Scale Invariance in UAV Bird's Eye View Object Detection by Adaptive Resizing," in *2022 26th International Conference on Pattern Recognition (ICPR)*, ISSN: 2831-7475, Aug. 2022, pp. 3588–3594. DOI: 10.1109/ICPR56361.2022.9956122. [Online]. Available: https://ieeexplore.ieee.org/document/9956122 (visited on 09/22/2024).

[36] D. Du *et al.*, "The Unmanned Aerial Vehicle Benchmark: Object Detection and Tracking," en, in *Computer Vision – ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds., vol. 11214, Series Title: Lecture Notes in Computer Science, Cham: Springer International Publishing, 2018, pp. 375–391, ISBN: 978-3-030-01248-9 978-3-030-01249-6. DOI: 10.1007/978-3-030-01249-6_23. [Online]. Available: https://link.springer.com/10.1007/978-3-030-01249-6_23 (visited on 06/01/2024).

[37] G. Xia *et al.*, "DOTA: A Large-Scale Dataset for Object Detection in Aerial Images," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2018, pp. 3974–3983. DOI: 10.1109/CVPR.2018.00418. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/CVPR.2018.00418.

[38] E. Bondi *et al.*, "BIRDSAI: A Dataset for Detection and Tracking in Aerial Thermal Infrared Videos," en, in *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, Snowmass Village, CO, USA: IEEE, Mar. 2020, pp. 1736–1745, ISBN: 978-1-72816-553-0. DOI: 10.1109/WACV45572.2020.9093284. [Online]. Available: https://ieeexplore.ieee.org/document/9093284/ (visited on 06/01/2024).

[39] J. Suo, T. Wang, X. Zhang, H. Chen, W. Zhou, and W. Shi, "HIT-UAV: A high-altitude infrared thermal dataset for Unmanned Aerial Vehicle-based object detection," en, *Scientific Data*, vol. 10, no. 1, p. 227, Apr. 2023, ISSN: 2052-4463. DOI: 10.1038/s41597-023-02066-6. [Online]. Available: https://www.nature.com/articles/s41597-023-02066-6 (visited on 03/15/2024).

[40] Hussein Rady, "Classification of Multilayer Neural Networks Using Cross Entropy and Mean Square Errors," en, *Journal of the ACS Advances in Computer Science*, vol. 2, no. 2, pp. 29–48, Jun. 2008, ISSN: 2682-3578. DOI: 10.21608/asc.2008.148466. [Online]. Available: https://asc.journals.ekb.eg/article_148466.html (visited on 07/16/2024).

[41] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and Efficient Object Detection," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020, pp. 10 778–10 787. DOI: 10.1109/CVPR42600.2020.01079. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/CVPR42600.2020.01079.

[42] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes (VOC) Challenge," en, *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun. 2010, ISSN: 0920-5691, 1573-1405. DOI: 10.1007/s11263-009-0275-4. [Online]. Available: http://link.springer.com/10.1007/s11263-009-0275-4 (visited on 06/26/2024).

[43] K.-Y. Wong, *WongKinYiu/yolov7*, original-date: 2022-07-06T15:14:06Z, Jul. 2024. [Online]. Available: https://github.com/WongKinYiu/yolov7 (visited on 07/03/2024).

TRITA-EECS-EX 2023:0000
Stockholm, Sweden 2024

www.kth.se

€€€€ For DIVA €€€€

{
”Author1”: { ”Last name”: ”Ahrendt”,
”First name”: ”Philipp”,
”Local User Id”: ”u1v0n2c8”,
”E-mail”: ”pcah@kth.se”,
”organisation”: {”L1”: ”School of Electrical Engineering and Computer Science”,
}
},
”Cycle”: ”2”,
”Course code”: ”EA258X”,
”Credits”: ”30.0”,
”Degree1”: {”Educational program”: ”Master’s Programme, ICT Innovation, 120 credits”
,”programcode”: ”TIVNM”
,”Degree”: ”Both Degree of Master of Science in Engineering and Master’s degree”
,”subjectArea”: ”Electrical Engineering”
},
”Title”: {
”Main title”: ”Investigating Altitude-Adaptive Methods for Enhancing Small Object Detection on UAVs”,
”Language”: ”eng” },
”Alternative title”: {
”Main title”: ”Undersökning av höjdadaptiva metoder för förbättrad detektering av små objekt på UAVs”,
”Language”: ”swe”
},
”Supervisor1”: { ”Last name”: ”Conradt”,
”First name”: ”Jörg”,
”Local User Id”: ”u1w41vp7”,
”E-mail”: ”jconradt@kth.se”,
”organisation”: {”L1”: ”School of Electrical Engineering and Computer Science”,
”L2”: ”Computational Science and Technology” }
},
”Supervisor2”: { ”Last name”: ”Moletta”,
”First name”: ”Marco”,
”E-mail”: ”marco@floxrobotics.com”,
”Other organisation”: ”FLOX Robotics AB”
},
”Examiner1”: { ”Last name”: ”Kumar”,
”First name”: ”Arvind”,
”Local User Id”: ”u1vukjbl”,
”E-mail”: ”arvkumar@kth.se”,
”organisation”: {”L1”: ”School of Electrical Engineering and Computer Science”,
”L2”: ”Computer Science” }
},
”Cooperation”: { ”Partner_name”: ”FLOX Robotics”},
"National Subject Categories": "20201, 20206, 20207",
”Other information”: {”Year”: ”2024”, ”Number of pages”: ”1,67”},
”Copyrightleft”: ”copyright”,
”Series”: { ”Title of series”: ”TRITA-EECS-EX” , ”No. in series”: ”2023:0000” },
”Opponents”: { ”Name”: ”Dalim Wahby”},
”Presentation”: { ”Date”: ”2022-03-15 13:00”
,”Language”:”eng”
,”Room”: ”via Zoom https://kth-se.zoom.us/j/ddddddddddd”
,”Address”: ”Isafjordsgatan 22 (Kistagången 16)”
,”City”: ”Stockholm” },
"Number of lang instances": "2",
”Abstract[eng]”: €€€€

Computer vision is vital for the recent development of aerial vision-based applications, however, small object detection remains a challenge even for state-of-the-art models such as YOLOv7-tiny. Object detection from Unmanned Aerial Vehicles (UAVs) is particularly affected because of their sometimes high flight altitude leading to a greater amount of small objects to detect, while their limited storage and computing power capacity restricts the model complexity.
Contextual information such as the flight altitude is often readily available from onboard sensors and can serve as relevant prior knowledge to a neural network learning to predict the size, location, and class of objects in an image. Despite that, research in this direction is sparse and focuses on specific applications, demanding more general approaches.
This thesis investigates various methods to integrate altitude information into the learning process of an object detection network.
The focus of this work lies in analyzing the influence of the proposed methods on precision, recall, and mean average precision (mAP) for aerial datasets across different altitude levels and in general. We demonstrate that concatenating the input image with the altitude information or adding an auxiliary head that predicts the altitude from an image can help to slightly boost the performance. Furthermore, despite a reduced average precision, a dynamic loss based on altitude can offer more controlled fine-tuning of the model depending on the specific requirements of a UAV-based detection task.
However, our results also showed that the greatest benefits stemmed from the addition of a small object detection head and the removal of the largest head which is unrelated to flight altitude.

Nevertheless, this approach could potentially be further improved through one of the altitude adaptive methods since they are not mutually exclusive.
The proposed methods and benchmarks provide a foundation for future research in the area of altitude-aware models as well as validate research on UAV-optimized YOLO models.
Overall, this work provides an overview of how contextual information could be integrated into an existing object detection model and its effects on the training process and inference performance.

€€€€,
"Keywords[eng]": €€€€
Small Object Detection, Altitude adaptive, Unmanned Aerial Vehicles (UAV), YOLOv7-tiny €€€€,
"Abstract[swe]": €€€€

Datorseende är avgörande för den senaste utvecklingen av applikationer baserade på flygseende, men detektering av små objekt är fortfarande en utmaning även för toppmoderna modeller som YOLOv7-tiny.
Objektdetektering från obemannade flygfarkoster (UAVs) påverkas särskilt på grund av deras ibland höga flyghöjd, vilket leder till en större mängd små objekt att upptäcka, samtidigt som deras begränsade lagrings- och datorkapacitet begränsar modellens komplexitet.
Kontextuell information som flyghöjd är ofta lättillgänglig från sensorer ombord och kan fungera som relevant förkunskap för ett neuralt nätverk som lär sig att förutsäga storlek, plats och klass för objekt i en bild. Trots detta är forskningen i denna riktning sparsam och fokuserar på specifika applikationer, vilket kräver mer generella tillvägagångssätt.
I den här avhandlingen undersöks olika metoder för att integrera höjdinformation i inlärningsprocessen för ett nätverk för objektdetektering.
Fokus för detta arbete ligger i att analysera de föreslagna metodernas påverkan på precision, återkallande och genomsnittlig genomsnittlig precision (mAP) för flygdataset på olika höjdnivåer och i allmänhet.
Vi visar att om man sammankopplar indatabilden med höjdinformationen eller lägger till ett extra huvud som förutsäger höjden från en bild kan det bidra till att öka prestandan något.
Trots en minskad genomsnittlig precision kan en dynamisk förlust baserad på höjd dessutom erbjuda en mer kontrollerad finjustering av modellen beroende på de specifika kraven för en UAV-baserad detekteringsuppgift.
Våra resultat visade dock också att de största fördelarna härrörde från tillägget av ett detekteringshuvud för små objekt och borttagandet av det största huvudet som inte är relaterat till flyghöjden. Trots detta kan denna metod potentiellt förbättras ytterligare genom någon av de höjdadaptiva metoderna eftersom de inte utesluter varandra.
De föreslagna metoderna och riktmärkena utgör en grund för framtida forskning inom området höjdmedvetna modeller samt validerar forskning om UAV-optimerade YOLO-modeller.
Sammantaget ger detta arbete en översikt över hur kontextuell information kan integreras i en befintlig objektdetekteringsmodell och dess effekter på träningsprocessen och inferensprestanda.

€€€€,
"Keywords[swe]": €€€€
Detektering av små objekt, Höjdadaptiv, Obemannade Flygfarkoster (UAV), YOLOv7-tiny €€€€,
}

acronyms.tex

```
%%% Local Variables:
%%% mode: latex
%%% TeX-master: t
%%% End:
% The following command is used with glossaries-extra
\setabbreviationstyle[acronym]{long-short}
% The form of the entries in this file is \newacronym{label}{acronym}{phrase}
%                                     or \newacronym[options]{label}{acronym}{phrase}
% see "User Manual for glossaries.sty" for the  details about the options, one example is shown below
% note the specification of the long form plural in the line below
\newacronym[longplural={Debugging Information Entities}]{DIE}{DIE}{Debugging Information Entity}
%
% The following example also uses options
\newacronym[shortplural={OSes}, firstplural={operating systems (OSes)}]{OS}{OS}{operating system}

% note the use of a non-breaking dash in long text for the following acronym
\newacronym{IQL}{IQL}{Independent -QLearning}

% example of putting in a trademark on first expansion
\newacronym[first={NVIDIA OpenSHMEM Library (NVSHMEM\texttrademark)}]{NVSHMEM}{NVSHMEM}{NVIDIA OpenSHMEM Library}

\newacronym{KTH}{KTH}{KTH Royal Institute of Technology}


\newacronym{SIoU}{SIoU}{}
\newacronym{UAV}{UAV}{Unmanned Aerial Vehicle}
\newacronym{AI}{AI}{Artificial Intelligence}
\newacronym{IMU}{IMU}{Inertial Measurement Unit}

\newacronym{mAP}{mAP}{Mean Average Precision}
\newacronym{AP}{AP}{Average Precision}
\newacronym{FLOPS}{FLOPS}{Floating Point Operations}

\newacronym{CNN}{CNN}{Convolutional Neural Network}
\newacronym{FPN}{FPN}{Feature Pyramid Network}

\newacronym{GPS}{GPS}{Global Positioning System}
\newacronym{IoU}{IoU}{Intersection over Unit}

\newacronym{BCE}{BCE}{binary cross-entropy}




% Datasets
\newacronym{POG}{POG}{PeopleOnGrass}
```