CHALMERS
UNIVERSITY OF TECHNOLOGY

Drone Detection Using Deep Neural Networks and Semi-Supervised Learning

Alice Karlsson
Gustav Rosin

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2020

Master's thesis 2020

Drone Detection Using Deep Neural Networks and Semi-Supervised Learning

ALICE KARLSSON
GUSTAV ROSIN

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2020

Drone Detection Using
Deep Neural Networks
and Semi-Supervised Learning
ALICE KARLSSON
GUSTAV ROSIN

Supervisor: Lucas Brynte, Department of Electrical Engineering
Examiner: Fredrik Kahl, Department of Electrical Engineering

Master's Thesis 2020
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Drone in a rural area.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2020

Drone Detection Using
Deep Neural Networks
and Semi-Supervised Learning
ALICE KARLSSON
GUSTAV ROSIN
Department of Electrical Engineering
Chalmers University of Technology

Abstract

The usage of drones has in recent years increased for both civilian and military purposes. With their small size and tractability, weaponized drones pose a major threat and are difficult to detect and classify with modern equipment such as radar. Since drones share many similar features with other common objects such as birds in its operation space, radar struggle with accurate classification of drones. Another approach to detect drones, as proposed in this thesis, is to utilize a camera based deep-learning object detection algorithm to detect and classify drones. To utilize a deep-learning algorithm, extensive computer resources and a vast amount of annotated data are required. However the availability of resources and annotated data is often limited. This thesis optimizes and adapts a RetinaNet and implements temporal information using three different methods. The implementations of temporal information utilize a pre-trained backbone to minimize the demand of annotated data. Furthermore a semi-supervised learning framework is developed to enable the use of unannotaded data and background data. The framework generates annotations for unannotated data and thus expanding the amount of available data. The methods for integrating temporal information and the semi-supervised learning framework was evaluated against the same test data as other state-of-the-art algorithms. The results show that the proposed methods for integrating temporal information were not advantageous with regards to the AP-score. However, by incorporating the generated annotated data and background data, the performance of the algorithm vastly improved with regards to the F1-score. It could not outperform state-of-the-art methods, however the resulting framework has shown great promise of being able to be used as an annotation tool for unannotated data.

Keywords: Deep Learning, Object detection, RetinaNet, Temporal information, Detectron2, Semi-supervised learning.

Acknowledgements

We would like to express our sincere thanks to our supervisor Lucas Brynte who has helped us during our thesis and provided invaluable ideas and insights. We would also like to greatly thank Angelo Coluccia and the team over at SafeShore. Without the data provided by them, this master thesis would not have been possible. We would also like to thank our supervisors at Saab, Stefan Eriksson and Stefan Holmgren for interesting discussions and help with shaping our project. Furthermore, we would also like to thank Per Johansson at Saab for helping us organize as well as collect data. Lastly, we would like to thank Fredrik Kahl for being our examiner.

Alice Karlsson, Gothenburg, June 2020
Gustav Rosin, Gothenburg, June 2020

Contents

List of Figures **xiii**

List of Tables **xv**

1 Introduction **1**
1.1 Background . . . 1
1.2 Related Work . . . 1
1.3 Purpose . . . 2
1.4 Proposed Approach . . . 2
1.4.1 Integration of Temporal Information . . . 3
1.4.2 Use of Unannotated Data . . . 3
1.5 Scope and Limitations . . . 3
1.6 Tools and Equipment . . . 4
1.6.1 Google Colab . . . 4
1.6.2 PyTorch . . . 4
1.6.3 Detectron2 . . . 4
1.7 Contribution . . . 5
1.8 Report Outline . . . 5

2 Theory **7**
2.1 Neural Networks . . . 7
2.1.1 Activation Functions . . . 8
2.1.2 Loss Functions . . . 9
2.1.2.1 Cross-Entropy Loss . . . 9
2.1.2.2 Focal Loss . . . 9
2.1.2.3 Smooth L1-Loss . . . 10
2.1.3 Optimizers . . . 11
2.1.4 Convolutional Neural Networks . . . 13
2.1.5 Transfer Learning . . . 14
2.2 Object Detection . . . 14
2.2.1 Two-Stage Detection . . . 14
2.2.2 One-Stage Detection . . . 15
2.2.3 IoU . . . 15
2.2.4 Non-Maximum Suppression . . . 16
2.3 ResNet . . . 16
2.3.1 Vanishing and Exploding Gradient Problem . . . 16

2.3.2 Performance Degradation . . . 17
2.3.3 ResNet Architectures . . . 17
2.4 Feature Pyramid Network . . . 18
2.4.1 Bottom-up Pathway . . . 19
2.4.2 Top-down Pathway with Lateral Connections . . . 19
2.5 RetinaNet . . . 20
2.5.1 Classification Subnet . . . 21
2.5.2 Regression Subnet . . . 21
2.6 Evaluation . . . 21
2.6.1 Precision . . . 22
2.6.2 Recall . . . 22
2.6.3 F1-Score . . . 22
2.6.4 Average Precision . . . 22
2.7 Semi-Supervised Learning . . . 23

3 Methods **25**
3.1 Visualization and Analysis of Dataset . . . 25
3.1.1 Visualization of Data . . . 25
3.1.2 Analysis of Data . . . 29
3.2 Implementation of RetinaNet . . . 30
3.2.1 Overview of Implementation . . . 30
3.2.2 Anchor Placement in Implementation . . . 30
3.2.3 Sub-networks in the Implementation . . . 31
3.2.4 Anchor Matching with Ground Truth . . . 31
3.2.5 Training with Predictions and Ground Truths . . . 31
3.2.6 Inference . . . 32
3.3 Modification of RetinaNet . . . 32
3.3.1 Defining a Baseline for the RetinaNet . . . 32
3.3.2 Defining a Baseline for Training . . . 32
3.3.3 Utilizing Transfer Learning . . . 33
3.3.4 Inclusion of P2 . . . 33
3.3.5 Defining a New Baseline for the RetinaNet . . . 33
3.3.6 Anchor Placements . . . 33
3.3.7 Pruning of P6 and P7 . . . 33
3.3.8 Relaxation of IoU Thresholds . . . 34
3.3.9 Tuning of Inference Parameters . . . 34
3.3.10 Final RetinaNet . . . 34
3.4 Temporal Information . . . 35
3.4.1 Concatenation of Feature Maps . . . 35
3.4.2 Siamese Networks with Addition Merge . . . 35
3.4.3 Siamese Networks with Concatenation Merge . . . 36
3.5 Implementation of Semi-supervised Learning Framework . . . 36
3.5.1 Semi-supervised Learning Framework . . . 37
3.5.2 Evaluation of Generated Annotations . . . 37

4 Results **39**
4.1 Evaluation Metrics . . . 39

4.1.1 Evaluation Drone vs Bird Detection Challenge 39
4.1.2 Evaluation of Semi-Supervised Learning Framework 39
4.2 Results RetinaNet 40
4.2.1 Evaluation of Baseline 40
4.2.2 Utilizing Transfer Learning 40
4.2.3 Inclusion of P2 41
4.2.4 Definition of New Baseline 41
4.2.5 Anchor Placement 42
4.2.6 Pruning of P6 and P7 43
4.2.7 Relaxation of IoU Thresholds 43
4.2.8 Tuning of Inference Parameters 44
4.2.9 Results Drone vs Bird Detection Challenge 45
4.3 Results Temporal Information 46
4.3.1 Concatenation of Feature Maps 46
4.3.2 Siamese Networks with Addition Merge 46
4.3.3 Siamese Networks with Concatenation Merge 47
4.4 Results Semi-Supervised Learning Framework 47
4.4.1 Visual Inspection of the Semi-Supervised Learning Framework 48
4.4.2 Semi-Supervised Learning Framework with Regards to the Drone vs Bird Detection Challenge 51

5 Discussion **53**
5.1 Discussion of Results of RetinaNet 53
5.2 Discussion of Result Drone vs Bird Detection Challenge 54
5.3 Discussion of Results of Temporal Information 55
5.3.1 Concatenation of Feature Maps 55
5.3.2 Siamese Networks with Addition Merge 56
5.3.3 Siamese Networks with Concatenation 56
5.3.4 Utilizing Temporal Information 57
5.4 Discussion of the Semi-supervised Learning Framework 57
5.4.1 Visual Inspection 57
5.4.2 Semi-Supervised Learning Framework with Regards to the Drone vs Bird Detection Challenge 58
5.5 Future Work 59

6 Conclusion **61**

Bibliography **63**

A Appendix 1 **I**

List of Figures

2.1 Visualization of an artificial neuron and a biological neuron [13]. CC-BY 8
2.2 Visualization of different commonly used non-linear activation functions [15]. CC-BY 8
2.3 Comparison of focal and cross entropy loss. Here, γ is a parameter that is tuned by the user. The standard cross entropy loss is represented when $\gamma = 0$. From [7]. CC-BY 10
2.4 Illustration of L1-loss, L2-loss and smooth L1-loss. Note how derivative is undefined for L1-loss when the loss is 0. From [18]. CC-BY . . 11
2.5 Illustration of smooth L1-loss. When the loss falls below a certain threshold, it switches to L2-loss. From [19]. CC-BY 11
2.6 Illustration of gradient descent with one parameter, w. From [21]. CC-BY 12
2.7 Visualization of how an image is proccessed and classified in a convolutional nerural network [23]. CC-BY 14
2.8 Visual representation of IoU From [27]. CC-BY 15
2.9 A residual block. From [35]. CC-BY 17
2.10 Table of the layers in different ResNet architectures. From [36]. CC-BY 18
2.11 The SSD-architecture. The auxilary layers are added at the output of the backbone. From[38]. CC-BY 18
2.12 Example of FPN structure. From[39]. CC-BY 19
2.13 Example of FPN structure with ResNet. From[39]. CC-BY 20
2.14 An example of RetianNet. From [41]. CC-BY 21
2.15 An example of PR-curve. From [44]. CC-BY 23

3.1 Drone flying during the night with LED-lights and a dark background 26
3.2 Drone flying during the day with no lights on and a light background. 26
3.3 A drone flying in a park far away. 27
3.4 A drone flying in a field far away. 27
3.5 Original image containing one drone far away flying close to a bird. . 28
3.6 Zooming in on the drone, one can see the contours of the bird flying close. 28
3.7 Histogram illustrating the distribution of ground truth sizes. 29
3.8 Histogram illustrating the distribution of aspect ratios in the ground truth. 29

3.9 Illustration of how feature pixels relate to the input pixels. The low level feature map has a stride of 2 and the high level feature map has a stride of 4 [47]. CC-BY 30

4.1 Comparison of true positives, false positives and false negatives with the final RetinaNet and the networks presented in the Drone vs Bird Detection Challenge. 45
4.2 Comparison of predictions from the Saab-test dataset on the same image by the two networks. 49
4.3 Comparison of misclassifications between the first and second network on the same image from the Saab-test dataset. 50
4.4 Comparison of misclassifications of the first and second network. . . . 50

A.1 Visual results from video 1. The drone is detected, however a false positive also occur I
A.2 Closer inspection of the detected drone from video 1 I
A.3 Visual results from video 2. The two drones are detected, however three false positive also occur II
A.4 Closer inspection of the detected drones from video 2 II
A.5 Visual results from video 3. The drone is detected and the nearby bird is not classified as a drone. However there is also a false positive III
A.6 Closer inspection of the detected drone from video 3 III

List of Tables

4.1 The standard RetinaNet parameters used in this work. 40
4.2 Performance of the baseline. 40
4.3 Parameters for experimenting with which layer to freeze until. 40
4.4 Results of freezing the baseline at different levels. 41
4.5 Parameters for the experiments when including p2 and removing p6 and p7. 41
4.6 Results of including p2 compared to not having p2. 41
4.7 Parameters of the old baseline compared to the new baseline. 41
4.8 Comparison of results from the old baseline and the new baseline. . . 42
4.9 Experiments with anchor placement. These experiments have the specified anchors across all feature levels. 42
4.10 Comparison of results for the anchor placements. 42
4.11 Experiment with anchors placed on individual feature levels. 42
4.12 Result of placing individual anchors across each feature level. 42
4.13 Comparison of the new baseline and the new baseline with p6 and p7 removed. 43
4.14 Comparison of results of the new baseline and the new baseline with p6 and p7 removed. 43
4.15 Comparison of the new baseline and the new baseline with the IoU thresholds reduced. 43
4.16 Results of the new baseline and the new baseline with the IoU thresholds reduced. 44
4.17 Comparision of results after tuning the score threshold. 44
4.18 Comparison of results after tuning the NMS parameter. 44
4.19 Comparison of results after tuning the topK parameter. 44
4.20 A comparison of the highest AP-scores achieved for the three versions of the new baseline after tuning the inference parameters. 44
4.21 Comparison of F1-scores from the Drone vs Bird Detection Challenge. 45
4.22 Results of concatenation of feature maps for the two time steps compared with the final RetinaNet. 46
4.23 Results of Siamese networks with addition merge for the two time steps compared to the final RetinaNet. 46
4.24 Results of concatenation merge for the two time steps at Res3. 47
4.25 Results of concatenation merge for the two time steps at Res4. 47
4.26 Amount of available data for the three datasets. 48
4.27 Comparison of the amount of data the two networks can annotate. . . 48

4.28 Comparison of amount of falsely annotated birds. 51
4.29 F1-score on the annotated Drone vs Bird Detection Challenge test data with the inclusion of generated annotations and background. . . 51

1
Introduction

This chapter will present the problems investigated in this master thesis, related work, and the contributions made. Lastly, an outline for the report is presented.

1.1 Background

In recent years, the availability of drones has increased. With their low price and easy access, accessibility has expanded beyond military powers to civilian users as well. Drones are difficult to detect even with fairly modern equipment due to their small size. Traditional methods such as radar struggle to separate drones from birds. One viable approach to overcome the problem of identification and detection is to use camera vision to detect and classify the drones.
In 2019 the 16-th IEEE International Conference on Advanced Video and Signal-based Surveillance (AVSS) held their annual Drone vs Bird Detection Challenge [1]. The goal of this challenge was the creation of a deep learning algorithm able to detect and classify drones in a video sequence where birds and motion in the fore and background may be present.
Often when training neural networks a vast amount of data is required. Since collecting and annotating said data can be onerous, the amount of data is often limited. This thesis proposes a method of identifying drones using a deep learning algorithm that maximizes the use of limited usable data by integrating temporal information. Additionally, a semi-supervised learning framework for incorporation of unannotated data will be investigated.

1.2 Related Work

In the Drone vs Bird Detection Challenge, different teams have created varying solutions to the challenge of drone detection. M. Nalamati et al. present a solution to the challenge using a Faster-RCNN with a base of ResNet-101 [2]. Another solution was to incorporate Super-Resolution techniques in order to increase the recall and thus increase the number of detected drones. This solution was presented by V. Magoulianitis et al. [3]. C. Craye et al won the competition 2019 and used a U-Net with ResNet110v2 and divided the detection and recognition paths into two networks [4]. D. Iglesia et al. presented a solution using a version of a RetinaNet in order to detect drones [5].
Outside of the Drone vs Bird Detection Challenge, further work has been made with the RetinaNet. C. Fu et al presented a modified version of a RetinaNet that ob-

tained a higher accuracy without increasing the computational cost. The presented approach adds mask predictions and a different loss function [6].
This master thesis utilizes a version of a RetinaNet. The original RetinaNet was proposed by T. Lin et al. The network consists of a backbone for feature extraction and two sub-networks for classification and regression. A loss function for the classification known as the focal loss was also proposed. This loss function aims to tackle the problem of heavy class imbalance between the foreground and the background. This resulted in the detector having the same accuracy as a two-stage detector but with the speed of a one-stage detector [7].
In order to incorporate temporal information into a neural network, G Sistu et al. proposed multi-stream fully convolutional networks. In their work a two stream FCN, a three stream FCN, and a network with two streams combined with an additional LSTM architecture were presented [8].
D. Chahyati et al. implemented a Siamese network based on a RetinaNet and incorporated the Hungarian algorithm for tracking humans in moving images [9].
X. Wang investigated the impact of different time steps for temporal information in a RetinaNet, and which combination is preferred in order to utilize temporal information [10].
Labeling a large amount of data can be very expensive when training a neural network, several methods exist that aim to reduce the amount of work needed. E. Sangineto et al. suggested a self-paced training protocol for object detection using only image level annotations. The region proposals of a Fast-RCNN were utilized to acquire proposal boxes and the box with the highest confidence score was marked as a pseudo-label. "Easy" examples were used in the early network in order to prevent the training from diverging and to reliably expand their training set [11].

1.3 Purpose

The purpose of this thesis is to detect and classify drones while simultaneously not classifying birds as drones. To accomplish this, this master thesis focuses on the study, development, and implementation of an object detection algorithm. The thesis also endeavors to improve an object detection algorithm without using large amounts of annotated data. Additionally, this thesis investigates a semi-supervised learning framework for the incorporation of unannotated data. It also aims to investigate whether the framework can be utilized to iteratively and reliably expand the amount of available annotated training data while utilizing unannotated data.

1.4 Proposed Approach

The project will start by modifying a standard RetinaNet to better fit the available data. Once complete, different strategies for incorporating temporal information and simultaneously utilize a pre-trained backbone will be developed and implemented. Finally, a semi-supervised learning framework for the incorporation of unannotated data will be created and evaluated. At the end of the thesis, the developed algorithm should be able to detect and classify drones in a setting occupied by birds.

The framework will be used to investigate whether the algorithm can be used to iteratively and reliably expand the amount of available annotated training data while only utilizing unannotated data. This framework is expected to be able to generate annotations for easy scenarios and be able to create a few annotations for more difficult scenarios.

1.4.1 Integration of Temporal Information

In object detection, the integration of temporal information has been proven to increase accuracy significantly. Temporal information refers to information from both space and time simultaneously. For example, in between consecutive frames of a video sequence, an object has moved in both space and time. The information in both of these frames is highly correlated and several methods exist that aim to utilize this information. A pre-trained backbone often improves the accuracy of a detection algorithm when only a small amount of training data is available. Since pre-trained backbones are not typically trained with the inclusion of temporal information, the parameters are only trained as feature extractors. Therefore, to utilize the pre-trained backbone, temporal information needs to be integrated in such a way that the pre-trained parameters are still used as feature extractors. For example in the work of [5], a frame difference channel was added to the three RGB-channels of the input image. Since the backbone is only trained with a three-channel input, this implementation could not utilize pre-trained parameters. This master thesis will study and evaluate different ways temporal information can be integrated, while also being able to utilize a pre-trained backbone. This will be accomplished by incorporating the difference between feature maps for consecutive timesteps into the neural network, both after and within the backbone.

1.4.2 Use of Unannotated Data

Annotated data is usually hard to find as well as expensive since it requires extensive manual labor. Therefore different ways to incorporate unannotated data will be investigated. This will be accomplished through the development of a semi-supervised learning framework. This framework will utilize the algorithm to generate annotations on unannotated examples.

1.5 Scope and Limitations

Due to time constraints, this master thesis will implement and evaluate three different methods of integrating temporal information. A RetinaNet will be used as a starting point and optimized to the best of our ability. The results of the optimized RetinaNet will be specifically tailored to the problem at hand and should not be seen as a general purpose solution. Due to the limited amount of data available, the methods used to improve the algorithm will utilize a pre-trained backbone. Limited access to GPU resources will also restrict how computationally expensive the suggested algorithm can be. The object detection algorithm will only be tuned for the classification of drones with other objects such as birds being regarded as

background. Only one version of the semi-supervised learning framework will be considered. The aim of this framework is to investigate whether the algorithm can be utilized as a simple annotation tool for drones. This investigation will be limited to a limited amount of unannotated collected data since the goal is to investigate the rough performance of the framework. The framework will further be limited to drones of roughly the same size as in the available annotated dataset from the Drone vs Bird Detection Challenge [1]. Furthermore, the evaluation of the framework will be preformed through visual inspection of a sample of generated annotations to determine if the annotations are comparable to hand annotated data. Further manual evaluation with regards to the confidence score will be made. Additionally, these annotations will be evaluated against the Drone vs Bird Detection Challenge test dataset. No additional hand-annotated data will be utilized to investigate the performance of the framework.

1.6 Tools and Equipment

This section presents the tools and equipment used in this Master thesis. The section begins with an introduction of Google Colab and PyTorch and lastly the object detection library Detectron2 is presented.

1.6.1 Google Colab

Google Colab is a service from Google that allows users to execute written Python code in a browser and provides free access to GPU resources. However, the amount of resources available varies day to day and is dependent on the current service usage and how much the user has recently used. The maximum continuous runtime is 12 hours and commonly available GPUs are Nvidia K80s, T4s, P4s, and P100s, however, there is not possible to choose which GPU one is assigned. The memory of the virtual machine also varies between runtimes, however, it does not vary during runtime [12].

1.6.2 PyTorch

PyTorch is a Python based open-source framework developed by Facebook's artificial-intelligence research group (FAIR). It can be utilized in conjunction with GPUs and is commonly used when developing deep learning projects. It has a simple and easy to use API making it user friendly.

1.6.3 Detectron2

The code was written in Python and used the platform Detectron2. Detectron2 is an open source PyTorch based modular object detection library created by Facebook's artificial-intelligence research group. State of the art object detection algorithms such as versions of Faster R-CNN and RetinaNet are implementable and easy to further modify thanks to Detectron2's modular design. Pre-trained model weights are

easily obtainable and usable. Furthermore, the datasets COCO, LVIS, CityScapes, and PascalVOC are integrated in Detectron2.

1.7 Contribution

The contribution of this thesis is an investigation of different methods for integrating temporal information with limited GPU resources and data for drone detection. The suggested semi-supervised learning framework has to the best of our knowledge, not been utilized for object detection before. Therefore the development and investigation of this framework will also be seen as a contribution.

1.8 Report Outline

Chapter 2 will cover the relevant theory in this thesis. The theory will cover the basics behind the neural network, object detection, the basic RetinaNet model as well as the evaluation metrics and Semi-supervised learning.

Chapter 3 will cover the methodology used in this thesis. It will present the available data, describe how the standard RetinaNet was implemented and modified, how the three different methods of temporal information were integrated, as well as a description of the semi-supervised learning framework.

Chapter 4 will cover the results of the Sections presented in the methodology as well as a more detailed description of the methods.

Chapter 5 will discuss the results in more detail and present suggested future work.

Chapter 6 will contain conclusions drawn from the results and discussion.

2
Theory

This chapter presents the relevant theory for this Master Thesis and aims to ease the understanding of the project. Section 2.1 covers the theoretical basics of artificial convolutional neural networks. Section 2.2 introduces object detection and IoU. ResNet will be presented in Section 2.3 and FPN in Section 2.4. RetinaNet is introduced in Section 2.5. Finally, different evaluations methods are presented in Section 2.6 and semi-supervised learning is introduced in Section 2.7.

2.1 Neural Networks

Neural networks are algorithms modeled to recognize patterns from an input with a design inspired by the neural network structure inside the human brain. A neuron consists of a cell body, an axiom, and dendrites. Most neurons receive an input signal via its dendrites and then produces an output signal through its axon. The information between two neurons is transferred via synapses, which enables the passing of an electrical or chemical signal to the other cell. An artificial neuron operates in much the same way as a biological neuron, see Figure 2.1. The input signal of an artificial neuron can be represented as x_i and the synapse is represented by the weight W_i. A neuron may have inputs from multiple sources, thus the i in these expressions represent each input. The information transferred into the dendrites of the target neuron can then be represented by $W_i x_i + b$ where b is the *bias*. In an artificial neural network, information is encoded by the frequency of which the neuron is sending information. The inputs to the neuron are summed and put through an activation function, that defines if the neuron should be considered to have sent information or not.

The artificial neural networks are able to learn from information because the parameter W_0 and b are trainable. Thus when receiving an input, W_0 and b should be selected so the information received is properly decoded. This process of selecting the proper value for W_0 and b is referred to as training. In a neural network, multiple neurons are used simultaneously. A group of neurons forms a layer and there are several layers in a complete network. The layers can be divided into the input layer, the hidden layers as well as the output layer. The input layer receives the initial input to the network and sends it through the hidden layers. The output of the network is produced after the hidden layers by the output layer. A standard neural network has all the neurons in the previous layers connected to all the neurons in the upcoming layer. This structure is known as fully connected layers [13].

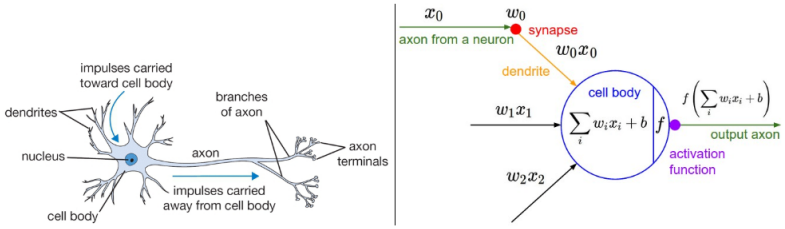


Figure 2.1: Visualization of an artificial neuron and a biological neuron [13]. CC-BY

2.1.1 Activation Functions

The activation function is responsible for deciding whether a neuron has sent information or not. The activation function is required to be non-linear to enable stacking of multiple layers and the usage of gradient based optimization methods. With a linear activation function, the stacked layers would simply be a linear combination of each other and can thus be replaced by a single layer. Furthermore, with gradient based optimization of the gradient with respect to the input would always be constant if a linear activation function was utilized. This leads to an optimization that is not dependent on the input [14]. The *ReLU* and the *Sigmoid* function are two of the most commonly used activation functions, see Figure 2.1.1. The *ReLU* function can be written as:

$$ReLU(x) = max(0, x). \tag{2.1}$$

The *Sigmoid* function can be written as:

$$Sigmoid(x) = \frac{1}{1 + e^{-x}}. \tag{2.2}$$

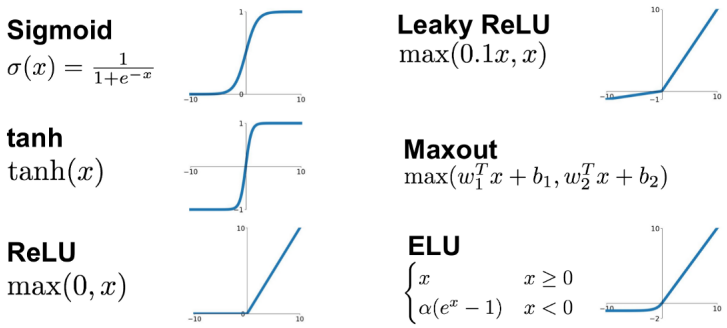


Figure 2.2: Visualization of different commonly used non-linear activation functions [15]. CC-BY

2.1.2 Loss Functions

Once an input has been given to a neural network and an output is received, one needs to calculate whether this output was correct or not. This accomplished by the inclusion of a loss function. The goal of the loss function is to minimize the error between the predicted output and the expected output. The expected output can be referred to as the ground truth. There are different kinds of loss functions adapted for solving specific problems, for example regression and classification problems.

2.1.2.1 Cross-Entropy Loss

The cross-entropy loss can be defined by either binary or multi-class cross entropy. Multi-class cross entropy can be split into multiple binary cross entropy functions were each function corresponds to one class of interest. In this thesis, binary cross entropy have been considered since only the drone class is of interest. The binary cross entropy loss can be written as:

$$CE(p, y) = \begin{cases} -log(p), & \text{if } y = 1 \\ -log(1-p), & \text{otherwise.} \end{cases}$$

In this equation, y is the ground truth and p is the estimated probability that the class has the label $y = 1$. The equation for binary cross entropy can be rewritten as:

$$CE(p_t) = -log(p_t). \tag{2.3}$$

Where p_t is defined as:

$$p_t = \begin{cases} p, & \text{if } y = 1 \\ 1-p, & \text{otherwise.} \end{cases}$$

The binary cross entropy loss receives its inputs from the last layer of the underlying neural network. The prediction is the estimated probability that the input was either foreground or background. Here, the foreground refers to the particular class that the binary problem corresponds to and the background refers to the other classes and no class [16].

2.1.2.2 Focal Loss

The focal loss is based on the commonly used cross entropy loss although it is designed to combat class imbalance, see Figure 2.3. Class imbalance, the imbalance between objects labeled as foreground and background, is a common issue. In a dataset there might be far more instances that are labeled as background instead of foreground. Should this be the case, the neural network might become overconfident in cases of background since the contribution of background is far larger than the contribution of foreground. This would result in poor performance when classifying a foreground example.
The focal loss aims to improve upon this problem by introducing a modulating factor to the cross entropy loss. The definition of focal loss can written as:

$$\mathrm{FL}\left(p_\mathrm{t}\right) = -\left(1-p_\mathrm{t}\right)^\gamma \log\left(p_\mathrm{t}\right). \tag{2.4}$$

The modulation factor $-(1-p_t)^\gamma$ is responsible for suppressing losses that have a large probability, because if a prediction is very accurate, the impact it will have on the loss should be considerably smaller compared to a prediction with a lot of uncertainty [7].

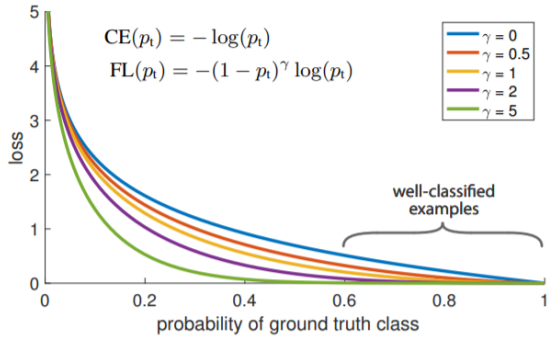


Figure 2.3: Comparison of focal and cross entropy loss. Here, γ is a parameter that is tuned by the user. The standard cross entropy loss is represented when $\gamma = 0$. From [7]. CC-BY

2.1.2.3 Smooth L1-Loss

Smooth L1-loss is a variant of the standard L1-loss that combines the properties of L1-loss when the loss is large and then switches to L2-loss as the loss gets smaller. The L1-loss can be written as:

$$S = \sum_{i=1}^{n} |y_i - f(x_i)| . \tag{2.5}$$

The L2-loss can be written as:

$$S = \sum_{i=1}^{n} (y_i - f(x_i))^2 . \tag{2.6}$$

In these equations, y_i is the ground truth and $f(x_i)$ is the approximated value. The L1-loss takes the absolute value of the difference between the target and the prediction while the L2-loss takes the square of the difference. The L1-loss is advantageous when the loss is large since it is robust to outliers and produces sparser solutions, however it is not differentiable when the loss is zero, see Figure 2.4. Thus the gradient-based optimization will be sub-optimal. Due to its quadratic nature, the L2-loss is differentiable when the loss is zero. The L2-loss produces more accurate results compared to the L1-loss because it penalizes larger errors to a higher extent, however, the L2-loss is more sensitive to outliers compared to the L1-loss [17].

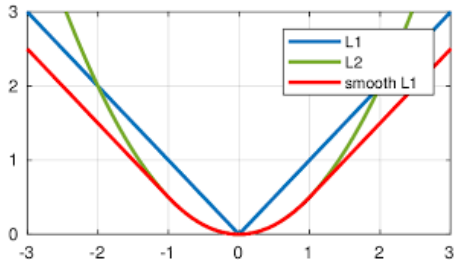


Figure 2.4: Illustration of L1-loss, L2-loss and smooth L1-loss. Note how derivative is undefined for L1-loss when the loss is 0. From [18]. CC-BY

The smooth L1-loss combines the L1 and L2-loss and uses the parameter β to distinguish between them. The smooth L1-loss can be written as:

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < \beta \\ |x| - 0.5 & \text{otherwise.} \end{cases} \tag{2.7}$$

When the loss is above a certain threshold decided by β, the regression loss will be L1-loss. When the loss falls below this threshold, the loss changes and instead becomes L2-loss, see Figure 2.5.

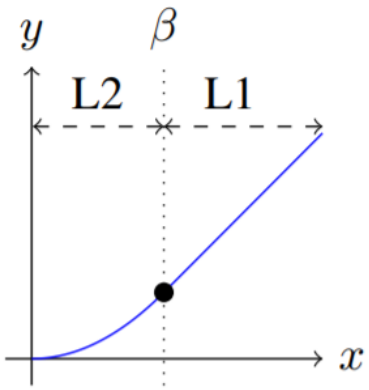


Figure 2.5: Illustration of smooth L1-loss. When the loss falls below a certain threshold, it switches to L2-loss. From [19]. CC-BY

2.1.3 Optimizers

Once the neural network has processed an input and the losses for the predictions are calculated, the trainable parameters need to be tuned in order to minimize the loss. This is done through an optimizer and the back-propagation algorithm.
It is common to use gradient based optimization methods, the most common of these methods is known as gradient descent, see Figure 2.6. In a neural network the amount of parameters depends on the number of layers, for notational simplicity, the weights and biases of all the layers are included in the parameter θ. The equation for gradient descent can be written as:

$$\theta^{t+1} = \theta^t - \alpha \frac{\partial E\left(X, \theta^t\right)}{\partial \theta}. \tag{2.8}$$

In this equation, α is the learning rate, which is a hyperparameter that indicates how much the weights are allowed to be updated during training. A too small α will result in an optimization that is very slow since the weights are updated by a small amount each pass. A too large α might result in the parameters changing too much and jumping over the optimum. $E(X, \theta^t)$ is the expected value for the loss function with network parameters θ at time t and input-output pairs: $(x_i, y_i) \in X$. This equation illustrates how the parameters in θ are updated by using the previous weights and the gradient of the loss function for those parameters given an input-output pair [20].

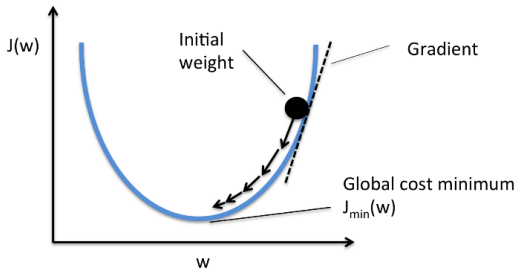


Figure 2.6: Illustration of gradient descent with one parameter, w. From [21]. CC-BY

In gradient descent, the weight update only occurs when the entire dataset has been processed. Due to the sheer amount of parameters in a neural network, the convergence of the optimizer to the global minimum may be very slow. Therefore an alternative form of gradient descent is utilized in neural networks. This form is known as stochastic gradient descent (SGD). The difference between gradient descent and stochastic gradient descent is that instead of processing the entire dataset before updating the weights, SGD takes a sample and updates the weights based on this sample.

Neural networks usually contain many layers which all contain a set of parameters. When using stochastic gradient descent to perform optimization, it is necessary to know how each of these parameters affects the final loss function. To calculate this, the back-propagation algorithm is used. Back-propagation consists of four parts: the forward pass, the loss function, the backward pass, and the weight update. The forward pass passes an input through the network to get a prediction. The loss function calculates the error of the prediction with respect to the expected output. The backward pass is then performed in order to relate how all the parameters in all layers contribute to the loss function. Once the relation between the parameters for the layers is known for the specific input, the weights are updated in such a way that the loss is reduced [22].

2.1.4 Convolutional Neural Networks

A convolutional neural network (CNN) includes neurons, activation functions, a loss function, and an optimizer but the layers differ from a basic neural network. A convolutional neural network utilizes convolutional layers instead of fully connected. The number of layers varies but the network consists of an input layer, multiple hidden layers, and an output layer. Convolutional layers are beneficial when dealing with images compared to standard fully connected layers since they utilize the concept of filters. An image can be thought of as a matrix with a specific width and height. In this matrix, each pixel is given a value based on its color gradient. Similarly, each filter is a matrix that also contains values, however, these values are trainable. If an image were to be processed in a fully connected fashion, the image matrix would need to be flattened into an array and each element would be assigned to a separate neuron as input. This would not only greatly increase the required parameters for the network, but the highly correlated information between adjacent pixels would not be utilized. Therefore, a convolution layer is more suitable for processing images compared to a fully connected one. A fully convolutional network is a convolutional network without any fully connected layers. These types of networks are very common in image segmentation.

Given an input image, the filters move as a sliding window across the pixels in the image. The amount of pixels that the filter is moved each iteration is specified by the stride. When the filter moves, the numerical values of the image are multiplied by the parameters inside the filter with element wise multiplication. These values are then summed up and the resulting value represents the information that was extracted by the filter at a specific location in the image. By applying filters to the inputs for the different layers, the network can differentiate and recognize the different object and features in images, see Figure 2.7.

The filters range from being able to detect basic features such as brightness in an image to more detailed features and characteristics of an object. Each layer usually contains more than one filter since each filter will only be trained to recognize a specific feature. The number of filters in a convolutional layer corresponds to the number of channels. Once the filter has slid or convolved over the entire image, the output is an array of numbers. This array is referred to as a feature map.

In a convolutional network maxpooling layers are added to reduce the spatial size to reduce the computational expense. This by dividing the output of the layers into regions and only keep the regions with the highest value, thus reducing the size. The intuition behind this is that the pixel of a feature map that has the highest value contains the most useful information, therefore the other adjacent pixels can be discarded [23].

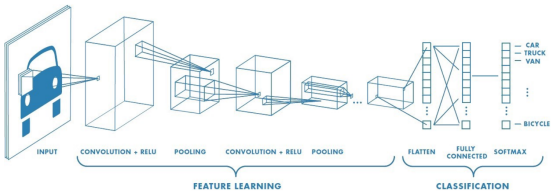


Figure 2.7: Visualization of how an image is proccessed and classified in a convolutional nerural network [23]. CC-BY

2.1.5 Transfer Learning

Transfer learning is when a model, trained and adapted for a specific task, is reused as base for another assignment. To train a convolutional network from scratch requires a large amount of data, thus it is advantageous to utilize pre-trained models. The pre-trained models are often good at extracting common features because they are trained on large data set, with images of different kinds of objects. Even though the specific image class is missing from the pre-training, the basic features, such as basic shapes can be used and then fine-tune the network for the specific task. This can be done by retraining only the deepest layers on the specific data [24].

2.2 Object Detection

Object detection can be divided into two parts, object localization, and object classification. Localization is used to point out where objects are located in the image and classification is used to decide what type of objects are present in an image. There are two different types of object detection techniques, two-stage object detection, and one-stage object detection. The output of an object detection algorithm is usually a box that covers the object as well as the predicted class label of that box. These boxes are referred to as bounding boxes. This section will cover two-stage detectors, one-stage detectors, and explain IoU and non-maximum suppression and how they are used in training and evaluation.

2.2.1 Two-Stage Detection

Two-Stage Object detectors perform localization and classification of an object in two stages, R-CNN, Fast R-CNN and Faster R-CNN are examples of two-stage detectors. The first stage is to decide regions on the image in which objects of interest may be present, so called regions of interest. These regions can be generated in different ways such as utilizing the simple search algorithm or use a neural network known as a region proposal network. Once these regions are decided the second stage is performed by passing these regions into a neural network, performing object classification. In addition, regression is performed to fit the proposed regions closer to the actual object. Since the model needs to perform two passes over the image, this method is not very fast [25].

2.2.2 One-Stage Detection

YOLO, SSD, and RetinaNet are one-stage object detectors that combine localization and classification into one step. Therefore, one-stage detectors only require a single stage. The regions of interest in one-stage detectors are acquired through the use of anchor boxes. One can think of the anchor box as an initial guess of where an object might be present. These anchors are evenly distributed on the input image, and for each of the anchors, a prediction is made whether they contain an object of interest or not. If an anchor is deemed to contain an object with the help of a classifier, the placement of the anchor is fine-tuned by a regressor to fit the object more accurately. The placement and sizes of these anchors are decided beforehand by the user, and thus no region proposal pass is required since the regions to investigate are decided beforehand. This often makes the object detection faster but less accurate compared to the two-stage object detection [26].

2.2.3 IoU

The intersection over union (IoU) is a measurement of how much one box overlaps with another box, see Figure 2.8. This can be written as:

$$IoU(box1, box2) = \frac{|box1 \cap box2|}{|box1 \cup box2|}. \tag{2.9}$$

.

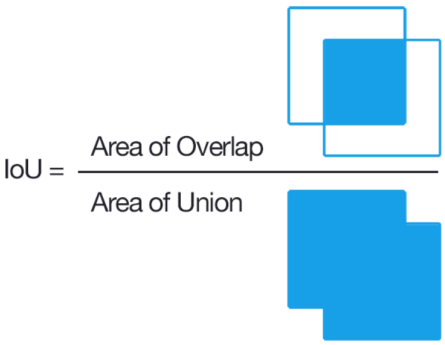


Figure 2.8: Visual representation of IoU From [27]. CC-BY

The IoU measurement is used to decide whether a proposed anchor box during training or predicted bounding box during evaluation is deemed to be correct. During training the model has access to all the ground truth bounding boxes and when the anchors are placed, the IoU is calculated for all the anchors and ground truths. If an anchor box has an IoU larger than the defined threshold over a ground truth, the model learns that this anchor contained an object with a specific class and how far away this anchor was from the ground truth. If the anchor box did not have an IoU over the threshold, the model learns that this anchor box just contained the background [28].
In the prediction step, anchor boxes are generated and a class is predicted for all anchor boxes. If an anchor box is deemed to contain an object, its position is

adjusted with regards to the predicted offset by the regressor in order to acquire the final bounding boxes used in prediction. The predicted bounding boxes are filtered with non-maximum suppression (NMS) in order to reduce multiple predictions of the same object [29].
When evaluating a model, the IoU is used to determine whether a predicted bounding box is correct or not. If a predicted bounding box is placed with an IoU larger than the threshold over a ground truth of the correct class, this prediction is considered to be correct. If a predicted bounding box does not have an IoU larger than the threshold over a ground truth, or the predicted class is incorrect, this prediction is considered to be incorrect. IoU on its own is not used to calculate any evaluation score, however, it is utilized in some common evaluation metrics.

2.2.4 Non-Maximum Suppression

To filter out multiple predictions of the same object, non-maximum suppression is conventionally used by object detectors. The algorithm compares all of the predictions confidence scores and the one with the highest score is selected. If any prediction has an IoU over some threshold with this prediction, it is discarded. This process is done for all the remaining predictions until no prediction has an IoU score over the threshold with another prediction [30].

2.3 ResNet

To improve the performance of neural networks, a common strategy is to add more layers and thus making it deeper. However, stacking more layers on top of another is problematic. When more layers are added, one encounters problems that are known as *Vanishing and exploding gradients* [31]. The problem with vanishing and exploding gradients has been addressed before ResNet. The problem was addressed in such a way that vanishing and exploding gradients did not hinder the network to converge from the beginning. A common way to combat this problem is to use the *ReLU* activation function since it has shown to be able to effectively handle this problem [32]. However, when deeper networks were evaluated, another problem manifested in the form of performance degradation. Even though it was possible to make the network converge from the start, an increase of layers surprisingly showed that both training and testing errors were increased with the addition of more layers. The purpose of ResNet was, therefore, to solve the vanishing and exploding gradient problem, while also taking care of the performance degradation problem by using skip connections to add the output from one layer to the other layers, skipping over some layers [33].

2.3.1 Vanishing and Exploding Gradient Problem

The problem with vanishing and exploding gradients occur during the backpropagation of the error function. During backpropagation, the gradient of the error is calculated backward through the network to tune the parameters of the network such that the loss function is minimized. Backpropagation is calculated by utilizing

the chain rule to represent how each layer affects the final loss function [34]. Sometimes, however, the gradients become very large or very small. Since the chain rule multiplies the gradients of all the layers, this might cause the final product to be close to zero or very large. With a too large or too small gradient, the network is unable to gather useful information of how each layer affect the loss function, and thus it is unable to learn [32].

2.3.2 Performance Degradation

The performance degradation problem was exposed when the previously utilized methods for combating the vanishing and exploding gradients revealed that the training and testing errors increased with deeper networks. The reason for this was that the currently available solvers simply could not find the desired mapping between the layers. The problem can be solved by utilizing so called residual blocks, see Figure 2.9. These blocks use a *skip-connection* between the non-linear layers and reformulate the problem slightly. Previously the goal was to find the desired *underlaying mapping* between the stacked layers. This mapping can be described as $\mathcal{H}(x)$. Since this mapping proved to be hard for solvers to find, the stacked layers were recast to fit another mapping, namley the *residual mapping*. This mapping can be described as $\mathcal{F}(x) := \mathcal{H}(x) - x$. This equation is then rewritten as $\mathcal{H}(x) = \mathcal{F}(x) + x$. It is easier for the solvers to find $\mathcal{F}(x)$ rather than finding the unreferenced $\mathcal{H}(x)$, because $\mathcal{F}(x)$ generally has a small response and thus the identity mapping, x provides a strong precondition [33].

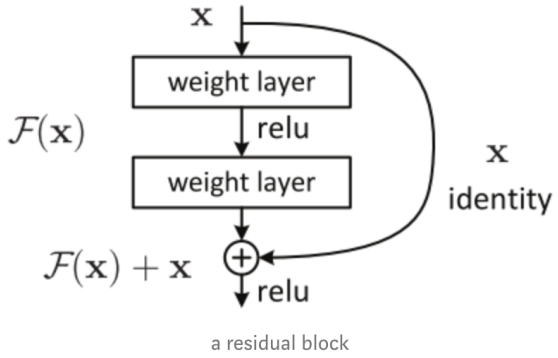


Figure 2.9: A residual block. From [35]. CC-BY

2.3.3 ResNet Architectures

The architectures of ResNet are based on stacking residual blocks on top of each other. ResNet-50, ResNet-101 and ResNet-152 are different kinds of ResNet architectures, where different numbers of layers are utilized. For example, ResNet-50 consists of 50 convolutional layers and ResNet-101 has 101, excluding the final fully connected layers.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				

Figure 2.10: Table of the layers in different ResNet architectures. From [36]. CC-BY

In a common ResNet each residual block consists of either two 3x3 layers or two 1x1 layers with a 3x3 layer in between, see Figure 2.10. The former of these two structures is called a *Bottleneck block*, however, the two structures work in the same way as a residual block. Each layer in Figure 2.10 is referenced as conv2_x up to conv5_x, however, for the remainder of this report, these layers will be referenced to as Res2 up to Res5.

2.4 Feature Pyramid Network

The concept of feature pyramids is used in many object detection algorithms, such as SSD, that utilizes a deep convolutional network in order to compute a feature hierarchy, see Figure 2.11. The deep convolutional network sub-samples layers which in turn create feature maps of varying spatial resolution. This inherent pyramidal feature hierarchy is used by the SSD in the form of auxiliary layers that are added on top of its backbone. These auxiliary layers then perform predictions on their own in conjunction with the output from the backbone. This enables the SSD to utilize multiple scales of spatial resolution when performing detections. However, a major downside comes with this approach, by adding these auxiliary layers to the output of the network the architecture does not utilize the higher resolution feature maps. The higher resolution feature maps do not have rich enough semantics since they have not been processed entirely by the network. Therefore it would not be beneficial to add them to the detection output [37]. By omitting these feature maps, the SSD struggles with the detection of small objects.

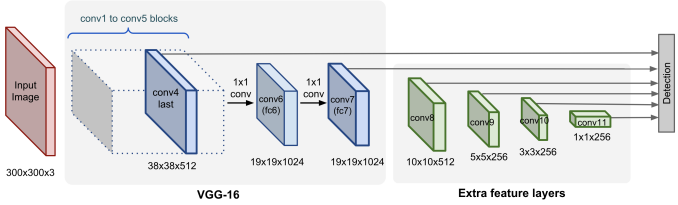


Figure 2.11: The SSD-architecture. The auxilary layers are added at the output of the backbone. From[38]. CC-BY

The feature pyramid network (FPN) is an improvement of the shortcomings of the SSD-architecture. The FPN architecture enables the use of high resolution feature maps that is simultaneous semantically strong. To achieve this, the architecture consists of a bottom-up pathway for feature extraction and a top-way pathway for up-sampling low resolution feature maps, see Figure 2.12. These feature maps are then combined through the use of lateral connections.

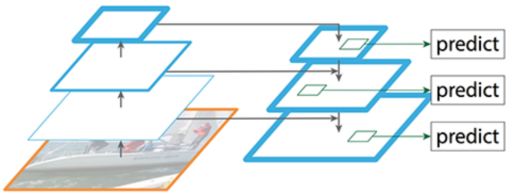


Figure 2.12: Example of FPN structure. From[39]. CC-BY

2.4.1 Bottom-up Pathway

The bottom-up pathway consists of a backbone that is used as a feature extractor. This backbone can, for example, consist of the feed forward computations of a ResNet, see Figure 2.13. At each level the feature map is sub-sampled with a factor of two and thus its dimensions are halved. This is represented by the different strides of the feature levels. The strides at different feature levels are $\{4, 8, 16, 32\}$ at $\{Res2, Res3, Res4, Res5\}$ respectively. The receptive field specifies the region of an image that is visible at each feature level, once the stride increases the receptive field increases. Since the receptive field increases with higher levels, these levels are more suitable for detecting larger objects. At these levels, the feature map is semantically strong due to its many convolutions, however, this comes with the cost that the spatial resolution is low. On the lower levels, the feature map has a higher spatial resolution and a smaller receptive field and is therefore more suitable for detecting smaller objects.

2.4.2 Top-down Pathway with Lateral Connections

In the top-down pathway, feature maps are merged in order to create feature maps that are both semantically strong and have high resolution. This is done by up-sampling the feature maps by a factor of 2 for each of the levels in the FPN. Nearest neighbor up-sampling is utilized and the up-sampled feature maps are then enhanced with the feature map of the corresponding size from the bottom-up pathway. The feature map from the bottom-up pathway lacks the strong semantics compared to the up-sampled ones from the top-down pathway, however they have higher resolution. Through the use of lateral connections between the two pathways, the semantically strong and high resolution feature maps are merged. The lateral connections is a convolutional layer with a kernel size of 1×1 and are used to set the channel depth to 256 for all feature maps across all layers from the bottom-up pathway. The feature maps are merged using element-wise addition and to reduce aliasing from the upsampling, each merged feature map is put through a final convolutional

layer with a kernel size of 3×3. After this process, each feature map have the strongest semantics possible since they all originates from the deepest layer and their resolution has been enhanced with the corresponding feature map from the bottom-up pathway [40].

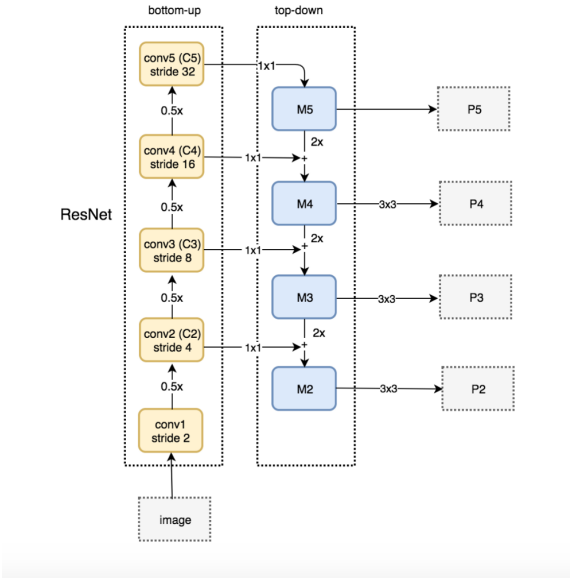


Figure 2.13: Example of FPN structure with ResNet. From[39]. CC-BY

2.5 RetinaNet

RetinaNet is an algorithm heavily based on the FPN architecture, it uses the FPN as its backbone but makes slight modifications to it, see Figure 2.14. The original FPN-backbone has feature levels $\{p2, p3, p4, p5\}$ that are related to the corresponding feed forward computations from the ResNet backbone $\{Res2, Res3, Res4, Res5\}$. The RetinaNet however elects not to use $p2$ as it is rather computationally expensive. Furthermore it includes two additional feature levels and thus making the RetinaNet consist of $\{p3, p4, p5, p6, p7\}$. The last two feature levels are based on the output from $Res5$, $p6$ is obtained by performing a convolution on $Res5$ with a kernel size of 3×3 and with a stride of 2. The last level, $p7$ is obtained by applying a convolution on $p6$ with a kernel size of 3×3 and a stride of 2. By including these two additional levels the detection of large object is improved [7]. Similarly to the FPN architecture, RetinaNet attached sub-networks to each feature level in order to make predictions. These sub-networks are designed for classification as well as regression and one of each is attached in parallel to each feature level.

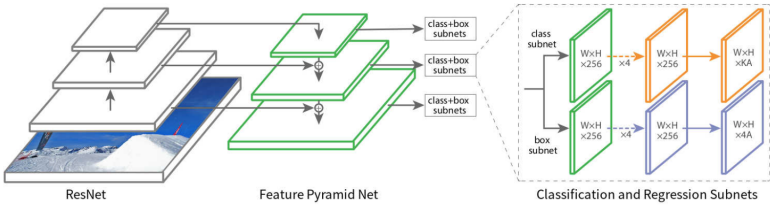


Figure 2.14: An example of RetianNet. From [41]. CC-BY

2.5.1 Classification Subnet

Identical classification sub-networks are placed on each feature level and all of them share parameters and consist of five fully convolutional layers with a kernel size of 3×3. The channel depth for the first four layers is 256, corresponding to the channel depth of the FPN output. The fifth layer has a channel depth of $A \times K$ where K is the number of classes to predict and A is the number of anchors for each spatial location. Between the first four layers there is a *ReLU* activation function and the output of the final layer is passed through a sigmoid activation function to obtain the binary predictions. The size of the last layer can be written as $(N, A \times K, H_i, W_i)$, N represent the batch size, H_i and W_i represent the height and width of the input feature map where i indicates the FPN level. The number of channels on the last layer corresponds to $A \times K$, thus for each spatial location A amount of anchors is defined. Each of these anchors is responsible for detecting any of the K available classes. Thus each output channel represents the probability of an anchor containing any of the classes. In the classification part of the RetinaNet, the focal loss function is utilized.

2.5.2 Regression Subnet

The classification sub-network is responsible for evaluating each anchor in order to decide if they contain an object or not, whereas the regression sub-network is responsible for deciding how much the anchor box should be shifted to more accurately fit the object. The regression sub-network is almost identical to the classification sub-network with five fully convolutional layers, however, the final layer differs slightly. The regression sub-network has a channel depth of $A \times 4$, four values that need to be predicted to fit the edges of the anchor box closer to the object, and the size of the output layer is $(N, A \times 4, H_i, W_i)$ [7].
The L1-loss function is used for the bounding box regression in the original implementation of the RetinaNet, however, in this work a smooth L1-loss is used instead. The smooth L1-loss combines the positive properties of both L1 and L2 loss and thus contributes to a better approximation of the bounding box regression.

2.6 Evaluation

This section covers two different ways of how to evaluate the performance of an object detector. This thesis will use F1-score and AP to evaluate the results.

2.6.1 Precision

Precision is a measurement of how well the model is able to only detect objects of interest. For example, if a model for detecting diseases have low precision, it will yield a high amount of positive detections even if only a few have the disease. The precision of an object detector can be calculated as:

$$Precision = \frac{TP}{TP + FP}. \tag{2.10}$$

TP is the true positives, the detections that were deemed correct and FP stands for false positive. IoU is utilized to determine the nature of the detection. If the predicted bounding box has an IoU larger than the threshold over a ground truth, the detection is deemed to be related to this ground truth. If the predicted class of the bounding box is the same as the class of the ground truth, the prediction is deemed to be correct. Otherwise, the detection is labeled as a false positive. Multiple predictions of the same object are also deemed to be false positives [42].

2.6.2 Recall

Recall is a measurement of how well the model is able to detect all objects of interest. A model for detecting disease in humans with a low recall will result in a lot of missed diagnoses where people are told they are healthy even though they are sick. The recall can be written as:

$$Recall = \frac{TP}{TP + FN}. \tag{2.11}$$

FN stands for false negative, the number of ground truths undetected by the predictor. If a predictor outputs a set of bounding boxes and none of the bounding boxes have an IoU over the ground truth that is larger than the threshold, this indicates that the model was not able to detect this object. Thus it is referred to as a false negative [42].

2.6.3 F1-Score

The F1-score combines both of these metrics to evaluate how well the model is able to prevent false detections and not miss actual detections [43]. The F1 score is the harmonic mean of precision and recall, thus both values are taken into consideration. The F1-score is defined as:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}. \tag{2.12}$$

2.6.4 Average Precision

In object detection problems, calculating the mean average precision is a common way to evaluate the performance of an algorithm. Average precision is the area under the curve of a precision-recall curve,

$$AP = \int_0^1 p(r)dr. \tag{2.13}$$

This curve plots values of its recall and precision on its x and y-axis correspondingly, see Figure 2.15. The precision and recall pairs are calculated for a confidence score threshold that is slowly decreasing. This threshold is taken from the confidence of the predictions, where highly confident predictions are investigated first and then as the threshold decreases, more of the not so confident predictions are considered.

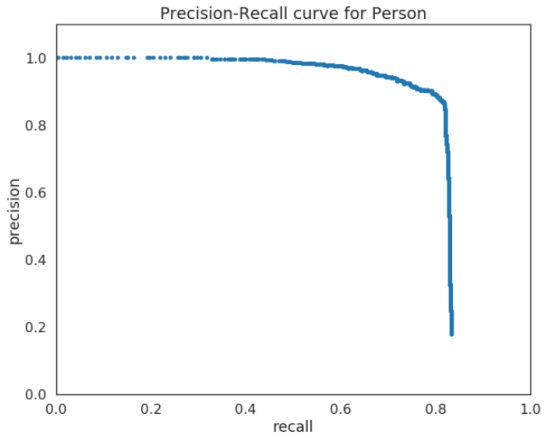


Figure 2.15: An example of PR-curve. From [44]. CC-BY

As the confidence score is decreased, lower quality predictions are allowed. This results in an increase in recall since more predictions are made and thus the number of false negatives is decreased. However, the precision is reduced when lower quality matches are allowed since more false positives occur with a lower confidence score. Since precision is reduced with increased recall, the maximum AP-score is achieved with a compromise between these two metrics. Mean average precision is simply the average of the average precision for all the available classes. In this work, only one class of drones has been considered and therefore average precision and mean average precision is the same [42].

2.7 Semi-Supervised Learning

Supervised learning methods refer to methods that utilize fully labeled datasets and aim to approximate a mapping function that given input, produces the desired output. This process is utilizing the labeled data as a reference during training to assert to the model whether a prediction was correct or not. Unsupervised learning does not have access to a labeled dataset and thus cannot utilize a reference during training. The aim of an unsupervised learning method is to find a relationship describing the structure in the data. An example of this is k-means clustering that aims to assign data points into one of K groups, to partition similar data points into the same category [45].
Semi-Supervised learning is a combination of supervised learning and unsupervised learning. It is a methodology for utilizing a small amount of annotated data and a large amount of unannotated data. One way to implement Semi-Supervised learning is by utilizing pseudo-labeling. First the network is trained in a supervised fashion,

using annotated data. Thereafter the model is used on unannotated data and generates predictions on this data. The prediction with the highest confidence score is considered to represent the true label and is thereafter utilized as annotated data [46].

3
Methods

This chapter will present the methods utilized to fit the existing RetinaNet architecture to suit the problem of small drone detection. Furthermore, the work done with temporal information and the semi-supervised learning framework will be presented. Section 3.1 will describe the analysis of the datasets from the Drone vs Bird Detection Challenge and provide a visualization of the training data with emphasis on the challenges to overcome. Section 3.2 presents the implementation of the RetinaNet. Section 3.3 will describe the modification of the RetinaNet. Section 3.4 describes how temporal information was incorporated and Section 3.5 presents the Semi-Supervised learning framework.

3.1 Visualization and Analysis of Dataset

The data set used for both training and evaluation of the algorithm comes from the competition Drone vs Bird Detection Challenge [1]. It contains eleven short video sequences for training and three short videos for testing. The images were taken with a static camera and contain both drones, birds, and background. However, only annotations for drones are available. The training data consists of 7245 images and the test data consists of 2079 images. The resolution of the images extracted was 1080×1920.

3.1.1 Visualization of Data

The data provided from the Drone vs Bird Detection Challenge depicts drones flying in a wide variety of different environments and contains challenging scenarios such as occlusion, moving objects in the background, flying within close proximity of birds, and a combination of these. As well as these challenges, the small size of the drones present an additional difficulty. The figures presented below illustrate a couple of samples of the data that were used for training the algorithm.

Drones are featured in different environments, see Figure 3.1 and Figure 3.2. In these figures the environment change with regards to the lightning of the background. One drone has clear LED-lights on and another has no lights on thus the appearance of the drone itself is changed. The environment can include land and urban features, see Figure 3.3 and Figure 3.4. The size of the drones was small and thus the algorithm needs to take into account both the change in the environment as well as the reduced size.

Figure 3.1: Drone flying during the night with LED-lights and a dark background

Figure 3.2: Drone flying during the day with no lights on and a light background.

Figure 3.3: A drone flying in a park far away.

Figure 3.4: A drone flying in a field far away.

Another challenge was the inclusion of birds flying next to the drones, see Figure 3.5 and Figure 3.6. The drone is far away and a bird is flying next to it. In this example, the algorithm needs to handle the small distant drone and simultaneously not misclassify the bird.

Figure 3.5: Original image containing one drone far away flying close to a bird.

Figure 3.6: Zooming in on the drone, one can see the contours of the bird flying close.

3.1.2 Analysis of Data

The majority of the annotations for the training data from the Drone vs Bird Detection Challenge have an area of 3-32 pixels, see Figure 3.7. The aspect ratios of the annotations were defined as the height of the annotation over the width, see Figure 3.8. It was most common for the width and height of the annotations to be equal, otherwise, the width is generally larger than the height.

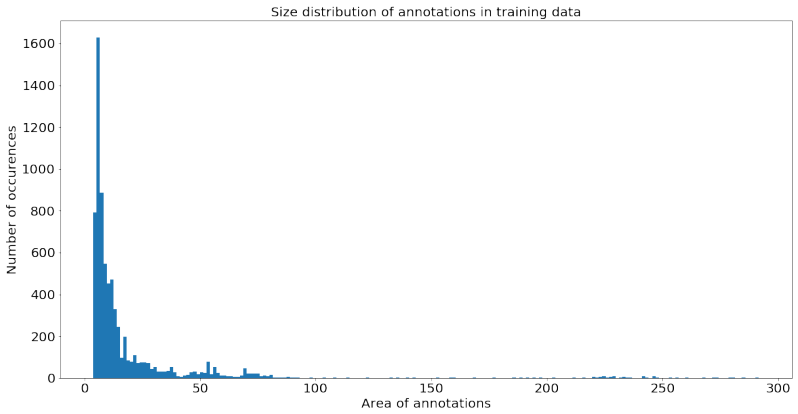


Figure 3.7: Histogram illustrating the distribution of ground truth sizes.

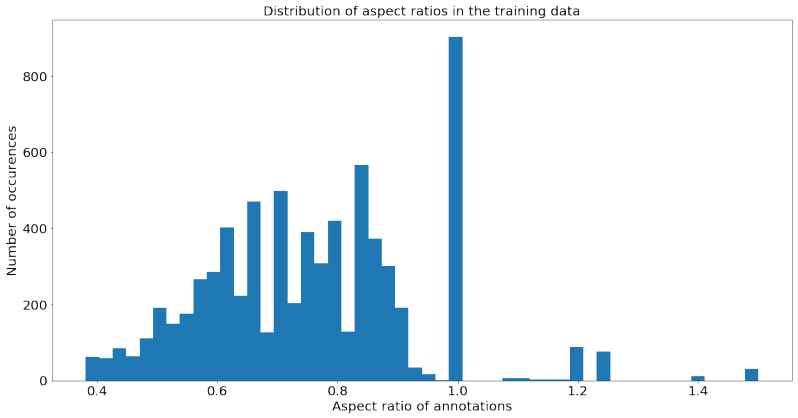


Figure 3.8: Histogram illustrating the distribution of aspect ratios in the ground truth.

3.2 Implementation of RetinaNet

The implementation of the RetinaNet as defined in [7] was found in the Detectron2 library. The following section aims to explain the standard implementation in detail.

3.2.1 Overview of Implementation

The standard RetinaNet, a part of Detectron2, is designed to be able to detect objects in a wide range of shapes and sizes. The RetinaNet is able to detect objects from eighty different classes and with sizes varying between 32-813 pixels [7]. The RetinaNet used in this thesis, before any modifications, consist of an FPN-backbone which has a bottom-up architecture of ResNet-101. The outputs of the bottom-up ResNet-101 consisted of $[Res3, Res4, Res5]$ and the corresponding output of the FPN was $[p3, p4, p5, p6, p7]$. The number of channels from the FPN was 256, and the features were merged with element-wise addition in the FPN. The pre-trained weights for ResNet-101 come from ImageNet MRSA R101.

3.2.2 Anchor Placement in Implementation

Anchors were placed on each feature level independently. In Detectron2 the anchors defined for each feature level were centered around each pixel on their corresponding feature map. The sizes of the feature maps were decided by the strides for each level. The strides for the outputs of the FPN were $[8, 16, 32, 64, 128]$, corresponding to level $[p3, p4, p5, p6, p7]$. These values indicate how much the input image was down-sampled at each level. The pixels on the feature map will henceforth be referred to as *Feature pixels* in order to avoid confusion from the pixels in the input image, also known as *Input pixels*. The anchors were centered around the feature pixels, on each feature map. Since ResNet is fully convolutional, the spatial positions on the feature maps can be related to the spatial position on the input, see Figure 3.9. Depending on the stride of the feature level, each feature pixel corresponds to an area that was defined as $Stride \times Stride$ on the input pixels. Once the feature pixels were related to the corresponding area on the input images, the anchor coordinates were defined with respect to their corresponding coordinates on the input image.

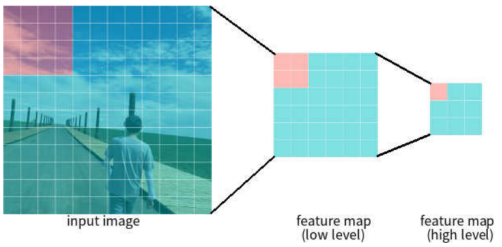


Figure 3.9: Illustration of how feature pixels relate to the input pixels. The low level feature map has a stride of 2 and the high level feature map has a stride of 4 [47]. CC-BY

3.2.3 Sub-networks in the Implementation

When the anchors were defined, the feature maps from the FPN-backbone was fed into the classification and regression sub-networks. In Detectron2, the output layer of the classification sub-network has channels corresponding to the number of anchors around each feature pixel times the number of classes that are to be predicted. This is represented as a tensor of size $(N, A \times K, H_i, W_i)$ were the index i denotes the feature level. The output of the regression sub-network is almost identical to the classifier output but the number of channels was defined as the number of anchors around each feature pixel times four. The four represent the edges of the anchor box and the values in these channels represent how much the edges should regress in order to fit the ground truth. This is represented by a tensor of size $(N, A \times 4, H_i, W_i)$. Given that an anchor was predicted to contain an object of interest, the goal of the regression sub-network was to decide how much this anchor box was to be regressed in order to fit the detected object properly.

3.2.4 Anchor Matching with Ground Truth

In order for the network to utilize a ground truth annotation, at least one of the anchors placed needs to overlap with the ground truth. Whether an anchor was considered a match with a ground truth was based on an IoU threshold. For a RetinaNet in Detectron2, anchor matching was performed individually for each anchor from all feature levels, for each of the ground truths in the image. If an anchor was labeled as foreground, it was assigned the correct label in the range [0, K-1] where K was the number of classes to predict. If it was labeled as the background it was assigned the label K. If the anchor was decided to be discarded, it was given the label -1. If an anchor was labeled to be the foreground, the distance for the four edges of this anchor to the ground truth was used as the ground truth for the regression sub-network.

3.2.5 Training with Predictions and Ground Truths

By matching the predictions from the sub-networks with the corresponding ground truth elements, the algorithm was trained. The predictions from the sub-networks were performed on each feature level of the image separately, thus needed to be interpolated and concatenated. Anchors with labels -1 were discarded. The classification loss was calculated with the focal loss, with regards to the predictions containing background and foreground. The focal loss parameter γ, was set to 2.0, and α to 0.25 to focus learning on hard negative examples. These values were taken from the work by T. Lin et. al, where the best combination was investigated [7]. The regression loss was calculated with the smooth L1 loss for anchors labeled as foreground. The β parameter in the smooth L1 loss was set to 0.1. Once the loss was between (-0.1,0.1) the L2 loss function was utilized. When the loss was outside the (-0.1, 0.1) range, L1 loss was utilized. Other values of the loss parameters were not investigated in this project.

3.2.6 Inference

During inference, the model utilized the predictions from the sub-networks and the pre-defined anchors. In Detectron2 inference was performed on each feature level independently and predictions on each feature level were concatenated and then put through non-maximum suppression. A Sigmoid function was used to receive the probability that the anchors contained an object of interest.
A filter was applied to the probabilities to only keep a certain amount of the top-scoring predictions. Another filter was applied to the remaining predictions that discarded predictions below a certain threshold. The anchor boxes remaining at this stage were fine tuned by their corresponding predictions from the regression sub-network. The predictions were concatenated and put through non-maximum suppression. The anchors left after non-maximum suppression was the final output from the model which contained both the bounding box coordinates for the prediction, the predicted class, and the confidence score of the bounding box.

3.3 Modification of RetinaNet

This Section will cover the different modifications of the RetinaNet to fit the model to the task of small object detection. At the end of this Section the final RetinaNet is introduced.

3.3.1 Defining a Baseline for the RetinaNet

The anchor sizes for the standard RetinaNet implemented by FAIR in Detectron2 needed to be resized to enable detection of smaller objects since the vast majority of training data from the Drone vs Bird Detection Challenge was below 32 pixels. The original implementation, by FAIR, placed anchors of different sizes on different feature levels however, in this work, the base RetinaNet has anchors of the same size placed on each feature map. The size of the input images were resized from 1080p down to 800x1333 to enable faster training.

3.3.2 Defining a Baseline for Training

The training for the experiments was conducted with 30.000 iterations due to limited GPU resources and based on the work by M. Nalamati et al [2]. The team utilized the dataset from the Drone vs Bird Detection Challenge and the same backbone of ResNet-101, as utilized by the RetinaNet in this thesis. It was discovered that the optimal training range was between 30.000 to 70.000 iterations. A linear warm up method with 1000 iteration was utilized. By linearly increasing the learning rate the influence of early training images was reduced, thus reducing the over-fitting in the early stages of training. However, in this project different warm-up parameters were not tested and evaluated. The warm up factor was 0.001 however, further experiments with the warm up factor for this project were not investigated.
A random horizontal flip of the training images was utilized as data augmentation. The probability of this flip occurring was 0.5.

Due to heavy class imbalance, a prior probability of 0.01 was utilized by the foreground class i.e the drone class in the classifier. The reason for this was that the background class tends to dominate the loss function and result in unstable training early on for datasets with heavy class imbalance [7]. Only one GPU was available thus the batch size was set to 2.

3.3.3 Utilizing Transfer Learning

Due to the small amount of available data from the Drone vs Bird Detection challenge, transfer learning was utilized. A study, investigating the impact of freezing the backbone at different layers, was performed. Four versions of the baseline RetinaNet were trained, for each of these networks, the backbone was frozen until Res1, Res2, Res3, and Res4.

3.3.4 Inclusion of P2

An additional feature level, p2 was added to the network thus enabling additional predictions on a feature map with higher resolution. Higher resolution feature maps are more suitable for detecting smaller objects, therefore an experiment was conducted to investigate whether the inclusion of an additional high resolution feature level would improve the accuracy.

3.3.5 Defining a New Baseline for the RetinaNet

A new baseline was defined where the input sizes were not resized. Furthermore, the standard aspect ratios were changed to better fit the available data.

3.3.6 Anchor Placements

Anchors of different sizes are often placed on different feature levels for feature pyramids such as FPN. However, this strategy advises a thorough study of the receptive field of each level. Due to time constraints, this study has been omitted, instead, experiments with anchor placements were performed. Different sizes, amounts, and placement of anchors were investigated.

3.3.7 Pruning of P6 and P7

In the feature pyramid architecture predictions were made on each feature level individually. The top levels p6 and p7 are semantically strong however the resolution of these feature maps is low. Lower resolution feature maps tend to struggle with accurate detections of smaller objects [5]. An experiment was conducted to investigate whether these levels could be removed to reduce the computational expense without reducing accuracy.

3.3.8 Relaxation of IoU Thresholds

In the standard RetinaNet, implemented by FAIR, an anchor box is considered a match with an annotation if the IoU is 0.5 or higher. For small objects, this threshold may be too strict. This would result in fewer or no anchors matching the ground truth and thus further increase the class imbalance. An experiment was conducted to investigate how a more relaxed threshold would affect performance, the IoU threshold was reduced from 0.5 down to 0.4.

3.3.9 Tuning of Inference Parameters

During inference, three parameters can be tuned to increase the performance of the algorithm. The parameters are TopK, Non-maximum suppression, and a score threshold. The TopK parameter specifies the maximum amount of predictions allowed by the network, three Topk values were evaluated. By default, the network keeps only the top 1000 scoring results. Non-maximum suppression was used to filter out multiple predictions of the same object. Four different values of the IoU threshold were evaluated. By default, the network has an IoU threshold of 0.5. The score threshold removes predictions with a confidence score smaller than some threshold. Four different values of this threshold were investigated. By default, the lowest allowed confidence score is 0.05. Due to time constraints, these parameters were only experimented with for a subset of the trained networks. The networks that were experimented with were the new baseline, the network with reduced IoU-threshold, and the network with p6 and p7 removed.

The reason for experimenting with the network with reduced IoU threshold comes from the fact that more anchors will be matched with the ground truth. Since anchors of the same size are placed on each feature level, this might cause an increase in false positives since there might be matches of the same object across different levels. Should this be the case, the NMS threshold during inference might be able to filter out these double matches.

It is also interesting to investigate the network with p6 and p7 removed. The reason for this comes from the fact that when removing p6 and p7 the performance is expected to decrease somewhat. However, this will make the network lighter and faster. With experiments of the inference parameters, it might be possible to reduce this decrease in performance.

3.3.10 Final RetinaNet

Once the optimal parameters for anchor placements, pruning of p6 and p7, relaxation of IoU thresholds and tuning of inference parameters were decided, they were implemented in the final RetinaNet. The final RetinaNet was used for the implementation of temporal information, and the semi-supervise learning framework.

3.4 Temporal Information

Once the final RetinaNet was chosen, methods for integrating temporal information were investigated. Due to the limited available data from the Drone vs Bird Detection Challenge, the suggested methods required to be compatible with the pre-trained backbone. The parameters of the pre-trained backbone have been trained as feature extractors of images. Therefore to utilize the pre-training, the methods needed to be integrated to enable the utilization of the pre-trained parameters as feature extractors in roughly the same way as they were trained. Three methods of integrating temporal information were implemented and evaluated. Each method was based on integrating information from one frame at time t, with the information from a previous time step. Two time steps were investigated, $t-1$ and $t-4$.

3.4.1 Concatenation of Feature Maps

This method utilized two identical copies of the FPN backbone. One backbone was given an input image of a frame at time t and the other one was given an input image of a previous frame. The backbone computed the feature maps for the two images and the feature maps from t were concatenated with the difference between the feature maps at time t and the previous time step. The concatenation was performed along the channels of the feature maps. The approach was inspired by the work of C. Craye et al [4]. The team concatenated grayscaled images of different time steps at the input to the network. Instead of performing the concatenation at the input, the concatenation was performed after the backbone. The additional channels contributed with additional information to the classifier and regression head, regarding the change of the feature map between frames.

Let $\mathcal{P}$ denote the set of feature maps from the backbone. Here $\mathcal{P}$ can be described by:

$$\mathcal{P} = \{p3,\ p4,\ p5,\ p6,\ \ p7\} \tag{3.1}$$

The concatenation of feature maps was done for two sets of feature maps, from time step t and the difference between the feature maps at time step t and $t-i$ where $i \in \{1, 4\}$. The set of difference between feature maps is denoted by:

$$\mathcal{P}_{sub} = \mathcal{P}_t - \mathcal{P}_{t-i} \tag{3.2}$$

The concatenation was performed on the feature maps in $\mathcal{P}$ and $\mathcal{P}_{sub}$ and can be written as:

$$\mathcal{P} \| \mathcal{P}_{sub} = \{p3 \,\|\, p3_{sub},\ p4 \,\|\, p4_{sub}\ , \ldots, p7 \,\|\, p7_{sub}\} \tag{3.3}$$

3.4.2 Siamese Networks with Addition Merge

An implementation of a Siamese network was made. The goal was to utilize the frozen layers as shared parameters between two consecutive frames. These frames were merged before the gradient enabled layers. This method aimed to be a more lightweight approach to integrating temporal information compared to the method

of concatenating the feature maps. With this method, only one set of model parameters needed to be stored in the memory. Furthermore, instead of having to save two full sets of feature maps, only one full set and one additional *Res3* was required. Backpropagation was only needed to be performed on one set of trainable layers.

Let $\mathcal{R}$ represent the set of feature maps from the backbone. This set can be divided into two subsets with one set representing the feature maps from the frozen layers and the other represents feature maps from the trainable layers. These sets can be written as $\mathcal{R}_{Frozen}$ and $\mathcal{R}_{Trainable}$ respectively they are described as:

$$\mathcal{R}_{Frozen} = \{Res1,\ Res2,\ Res3\} \tag{3.4}$$

$$\mathcal{R}_{Trainable} = \{Res4,\ Res5\} \tag{3.5}$$

The proposed methodology was to feed two consecutive frames through the layers in $\mathcal{R}_{Frozen}$ and then merge the feature maps before the feature maps were fed into the layers in $\mathcal{R}_{Trainable}$. The merge was performed by adding the feature maps and take the average by dividing by two. The succeeding layers were pre-trained and expected an input in the form of a feature map from an image. By receiving the average of the two feature maps, the input to the pre-trained parameters was similar to the expected input.

3.4.3 Siamese Networks with Concatenation Merge

The final proposed implementation of temporal information was a combination of the previous methods. In the same way as the previous Siamese network, the set of feature maps, $\mathcal{R}$ was divided into $\mathcal{R}_{Frozen}$ and $\mathcal{R}_{Trainable}$. Two consecutive frames were fed into the frozen layers. The merge was done by concatenating the feature map at time t with the difference between the feature map at time t and the feature map at time $t - i$. The choice of the merge was based on the common practice of merging through concatenation rather than addition, thus the difference between these two types of merge was investigated. The concatenation merge contributed with additional information regarding the change of the feature map between frames. An auxiliary convolutional layer with kernel size 1×1 was utilized to reduce the number of channels to the appropriate amount expected by the next coming layer. This method, in addition to merging at *Res3*, performed an experiment with merging at *Res4*. The reason for this was to investigate how the location of the merge in the network affected performance.

3.5 Implementation of Semi-supervised Learning Framework

The goal of the semi-supervised learning framework suggested in this thesis was to enable the use of unannotated data in a modern object detection algorithm. Since gathering data is considered rather easy at Saab, the difficult part would be to

manually annotate a vast amount of data. Therefore it was investigated whether the algorithm could reliably be utilized as an annotation tool for unlabeled data. The data collected for this thesis contains scenes that were largely different compared to the data found in the Drone vs Bird Detection Challenge datasets. This will challenge the generality of the algorithm since the collected data can be considered belonging to another domain. Furthermore, the data collected and annotated by the framework was incorporated into the Drone vs Bird Detection Challenge datasets, to investigate whether data from another domain than the Drone vs Bird Detection Challenge could be utilized to further improve the performance of the algorithm.

3.5.1 Semi-supervised Learning Framework

The semi-supervised learning framework draws inspiration from the work of E. Sangineto et al [11]. The team utilized a Fast-RCNN to generate bounding box annotations on images using the region proposal network in the first iteration and later utilizing the predicted bounding boxes as annotations. The annotations were performed on easily classified samples for the first iterations and more difficult classified examples were successively added between iterations. By utilizing easily classified samples in the first iterations the training was kept from diverging in the early stages due to noisy annotations. The framework in this thesis utilized easily classified samples of drones flying in an area that differs from the provided training data from the Drone vs Bird Detection Challenge. Predictions were made on the easy samples using the algorithm and if these samples proved to be comparable to hand annotated data, they were utilized as training data for the next iteration.

The collected data consisted of three parts, background, training and test data. The background dataset consisted of a scene where birds may be present, however, no drones were included thus the scene was annotated as background. The training data consisted of a video with a rather easy case where the drone was flying towards an area that was occupied with traditional Swedish houses as well as pine trees. The test dataset was similar to the training dataset, however, with more emphasis on difficult cases such as flying the drone around treetops and multiple cases of the drone flying in front of buildings.

3.5.2 Evaluation of Generated Annotations

The evaluation of the network trained with the generated annotations was made with visual inspection, investigation of confidence scores and misclassifications, and through evaluation against the Drone vs Bird Detection Challenge test data. The visual inspection was performed by sampling a section of the proposed annotations and an investigation whether the annotations were competitive to work done by human annotators was made. The quality of the annotations was investigated between iterations of the framework, by investigating the confidence scores of predictions made between iterations and how accurately the annotations were placed on the drone. Furthermore, misclassified objects were investigated between iterations, to investigate whether these misclassifications were reduced between iterations.

Furthermore, the number of frames that were deemed to be usable between training iterations was a factor when evaluating the performance of the framework.

Since only a sample of the generated annotations was evaluated through visual inspection, further evaluation of the quality of the annotations was needed. The performance of the network trained with the generated annotated dataset was evaluated with the Drone vs Bird Detection Challenge test data. The reason for this was to investigate whether the generated annotations could contribute to an increase in performance on any of the test videos from the Drone vs Bird Detection Challenge. The generated annotations might prove to contain noisy labels that were not discovered in the visual inspection. Therefore this evaluation was used to determine whether the benefits of the generated annotations outweigh the shortcomings. The F1-score was used as the evaluation metric.

4
Results

This chapter will give an in-depth explanation of the experiments conducted in the methodology and present the results of these experiments. Section 4.1 will present the evaluation metrics. In Section 4.2 the results regarding the modifications of the RetinaNet will be presented. Section 4.3 will present the results of the implementation of temporal information. In Section 4.4, the results from the semi-supervised learning framework will be covered. In this thesis, five different datasets have been used. Section 4.2 and 4.3 will only utilize the training and test data from the Drone vs Bird Detection Challenge. In Section 4.4, three new datasets will be introduced. These datasets will be utilized by the semi-supervised learning framework in conjunction with the training and test datasets from the Drone vs Bird Detection Challenge.

4.1 Evaluation Metrics

Two main evaluation metrics have been utilized to evaluate the performance, the COCO AP metric, and the F1-score. The results regarding the tuning of the RetinaNet and temporal information were evaluated with the implemented COCO AP function. The AP-score calculated for each evaluation was based on an average over multiple IoU-thresholds. Theses thresholds ranged from $0.50 : 0.95$ with an increment of 0.05 between each threshold. AP-score for small, medium, and large objects was obtained for each evaluation. An object was considered small if the area of its ground truth bounding box was smaller than 32 pixels, medium if it was between 32-96 pixels and finally large if it was larger than 96 pixels [48].

4.1.1 Evaluation Drone vs Bird Detection Challenge

In the Drone vs Bird Detection Challenge, another metric was utilized to evaluate the final algorithms, the F1-score. To compare the final algorithm with the state of the art algorithms, the F1-score of the final RetinaNet was evaluated using this metric.

4.1.2 Evaluation of Semi-Supervised Learning Framework

To evaluate the semi-supervised learning framework, visual inspection was utilized. Furthermore, the semi-supervised learning framework was evaluated with the F1-score to compare how the use of this framework improved the results in the Drone vs Bird Detection Challenge.

4.2 Results RetinaNet

This section presents the results of the original implemented baseline, how transfer learning was utilized, and the inclusion of P2. Thereafter a new baseline is introduced followed by the results of the anchor placement and the pruning of p6 and p7. Consequently, the results of the relaxation of the IoU thresholds and the tuning of inference parameters will be presented. Finally the results of the Drone vs Bird Detection Challenge will be presented. The datasets used in Section 4.2 is the training and test data from the Drone vs Bird Detection Challenge.

4.2.1 Evaluation of Baseline

The baseline parameters of the RetinaNet are the default settings from the RetinaNet implemented by FAIR, see Table 4.1. The anchor sizes were selected based on the sizes of the annotations in the Drone vs Bird Detection Challenge dataset. The initial input size was down-sampled from the native resolution of 1080 $\times$ 1920 to the size 800 $\times$ 1333. The AP-scores for this baseline were calculated, see Table 4.2.

Table 4.1: The standard RetinaNet parameters used in this work.

Frozen until	IoU	Feature levels	Aspect ratios	Anchors	Input size
Res4	[0.4, 0.5]	p3-p7	[0.5, 1.0, 2.0]	[4, 8, 16, 32, 64, 128, 256]	800 x 1333

Table 4.2: Performance of the baseline.

AP	AP(50)	AP-s	AP-m	AP-l
11.659	30.705	6.954	38.517	36.583

4.2.2 Utilizing Transfer Learning

In these experiments, all parameters were kept constant according to the baseline, and only the point where the backbone was frozen until was changed, see Table 4.3. These experiments indicated that to achieve the highest overall AP-score, the backbone should be frozen until *Res*3, see Table 4.4.

Table 4.3: Parameters for experimenting with which layer to freeze until.

	Frozen until	IoU	Feature levels	Aspect ratios	Anchors	Input size
Experiment 1	**Res1**	[0.4, 0.5]	p3-p7	[0.5, 1.0, 2.0]	[4, 8, 16, 32, 64, 128, 256]	800 x 1333
Experiment 2	**Res2**	[0.4, 0.5]	p3-p7	[0.5, 1.0, 2.0]	[4, 8, 16, 32, 64, 128, 256]	800 x 1333
Experiment 3	**Res3**	[0.4, 0.5]	p3-p7	[0.5, 1.0, 2.0]	[4, 8, 16, 32, 64, 128, 256]	800 x 1333
Experiment 4	**Res4**	[0.4, 0.5]	p3-p7	[0.5, 1.0, 2.0]	[4, 8, 16, 32, 64, 128, 256]	800 x 1333

Table 4.4: Results of freezing the baseline at different levels.

	AP	**AP(50)**	**AP-s**	**AP-m**	**AP-l**
Experiment 1	13.678	38.248	10.566	32.029	30.898
Experiment 2	13.221	31.392	6.231	43.556	38.677
Experiment 3	**14.583**	38.792	9.575	37.673	43.811
Experiment 4	11.659	30.705	6.954	38.517	36.583

4.2.3 Inclusion of P2

When including p2 it became apparent that this feature level was computationally expensive. To reduce the training time, the top feature levels p6 and p7 were removed. To get a fair comparison, an additional model with the same parameters as the baseline with p6 and p7 removed was trained, see Table 4.5. These experiments indicated that the inclusion of p2 would be beneficial, see Table 4.6. The inclusion of p2 improved the network's ability to detect smaller objects significantly. Even without p6 and p7, the overall AP-score was higher compared to the baseline RetinaNet. However, due to computational expense, it was elected to not include p2 in the final algorithm since the limited GPU-resources prohibited this.

Table 4.5: Parameters for the experiments when including p2 and removing p6 and p7.

	Frozen at	**IoU**	**Feature levels**	**Aspect ratios**	**Anchors**	**Input size**
Experiment 1	Res4	[0.4, 0.5]	**p2-p5**	[0.5, 1.0, 2.0]	[4, 8, 16, 32, 64, 128, 256]	800 x 1333
Experiment 2	Res4	[0.4, 0.5]	**p3-p5**	[0.5, 1.0, 2.0]	[4, 8, 16, 32, 64, 128, 256]	800 x 1333

Table 4.6: Results of including p2 compared to not having p2.

	AP	**AP(50)**	**AP-s**	**AP-m**	**AP-l**
Experiment 1	**11.977**	34.442	8.179	35.254	7.827
Experiment 2	9.661	21.369	2.218	46.143	35.525

4.2.4 Definition of New Baseline

A new baseline was evaluated where the image was not resized and the standard aspect ratios were changed. For this baseline, the number of frozen layers were based on the result of the experiments in Section 4.2.2. A comparison between the old baseline and the new baseline was made, see Table 4.7. A tremendous increase in accuracy was achieved for the new baseline. The AP-score for small and medium objects increased significantly since the higher resolution of the input image enabled the network to detect smaller objects with higher accuracy. Based on these results, it was elected to use the new baseline, see Table 4.8.

Table 4.7: Parameters of the old baseline compared to the new baseline.

	Frozen until	**IoU**	**Feature levels**	**Aspect ratios**	**Anchors**	**Input size**
Experiment 1	Res4	[0.4, 0.5]	p3-p7	[0.5, 1.0, 2.0]	[4, 8, 16, 32, 64, 128, 256]	800 x 1333
Experiment 2	**Res3**	[0.4, 0.5]	p3-p7	**[0.4, 0.7, 1.0]**	[4, 8, 16, 32, 64, 128, 256]	**1080 x 1920**

Table 4.8: Comparison of results from the old baseline and the new baseline.

	AP	**AP(50)**	**AP-s**	**AP-m**	**AP-l**
Experiment 1	11.659	30.705	6.954	38.517	36.583
Experiment 2	**22.009**	48.585	17.267	49.365	22.207

4.2.5 Anchor Placement

Experiments regarding the anchor placement were conducted for the new baseline, see Table 4.9 and Table 4.11. The experiments were mainly performed with the same anchors placed on all feature levels. However, one experiment was conducted with different sizes of anchors placed on each feature level individually. This experiment was inspired by the implementation of the original RetinaNet where a fixed anchor size is scaled with a vector $[2^0, 2^{\frac{1}{3}}, 2^{\frac{2}{3}}]$, see Table 4.11. Based on these results, it was elected to use anchors of size [4, 8, 16, 32, 64, 128, 256] placed across all feature levels, see Table 4.10 and Table 4.12.

Table 4.9: Experiments with anchor placement. These experiments have the specified anchors across all feature levels.

	Frozen until	**IoU**	**Feature levels**	**Aspect ratios**	**Anchors**	**Input size**
Experiment 1	Res3	[0.4, 0.5]	p3-p7	[0.4, 0.7, 1.0]	**[4, 8, 16, 32, 64, 128, 256]**	1080 x 1920
Experiment 2	Res3	[0.4, 0.5]	p3-p7	[0.4, 0.7, 1.0]	**[2, 4, 8, 16, 32, 64]**	1080 x 1920
Experiment 3	Res3	[0.4, 0.5]	p3-p7	[0.4, 0.7, 1.0]	**[2, 4, 8, 16, 32]**	1080 x 1920
Experiment 4	Res3	[0.4, 0.5]	p3-p7	[0.4, 0.7, 1.0]	**[4,8,16,32,64,128]**	1080 x 1920

Table 4.10: Comparison of results for the anchor placements.

	AP	**AP(50)**	**AP-s**	**AP-m**	**AP-l**
Experiment 1	**22.009**	48.585	17.267	49.365	22.207
Experiment 2	19.132	49.21	14.363	43.81	44.071
Experiment 3	21.195	49.50	16.011	53.452	42.845
Experiment 4	19.481	49.259	17.618	34.92	24.739

Table 4.11: Experiment with anchors placed on individual feature levels.

	P3	**P4**	**P5**	**P6**	**P7**
Experiment 5	$2 \times [2^0, 2^{\frac{1}{3}}, 2^{\frac{2}{3}}]$	$4 \times [2^0, 2^{\frac{1}{3}}, 2^{\frac{2}{3}}]$	$8 \times [2^0, 2^{\frac{1}{3}}, 2^{\frac{2}{3}}]$	$16 \times [2^0, 2^{\frac{1}{3}}, 2^{\frac{2}{3}}]$	$32 \times [2^0, 2^{\frac{1}{3}}, 2^{\frac{2}{3}}]$

Table 4.12: Result of placing individual anchors across each feature level.

	AP	**AP(50)**	**AP-s**	**AP-m**	**AP-l**
Experiment 5	16.064	30.761	6.585	59.985	63.767

4.2.6 Pruning of P6 and P7

A comparison of the new baseline and the new baseline with p6 and p7 removed was made, see Table 4.13. Based on the results of this experiment, it was elected to keep p6 and p7, see Table 4.14. When P6 and P7 were removed the overall AP-score decreased which was expected. The AP-score for small and medium objects stayed roughly the same which indicates that the majority of these objects were not dependent on p6 and p7 to be detected. However, the AP-score for larger objects increased when p6 and p7 were removed. Even though the model performed rather similar when including p6 and p7 versus removing them, the removal still resulted in a lower overall AP-score. Since the highest feature levels are the least computationally expensive, the levels were deemed to be worth the cost in computational expense for the ability to increase the overall AP-score.

Table 4.13: Comparison of the new baseline and the new baseline with p6 and p7 removed.

	Frozen until	**IoU**	**Feature levels**	**Aspect ratios**	**Anchors**	**Input size**
Experiment 1	Res3	[0.4, 0.5]	p3-p7	[0.4, 0.7, 1.0]	[4, 8, 16, 32, 64, 128, 256]	1080 x 1920
Experiment 2	Res3	[0.4, 0.5]	**p3-p5**	[0.4, 0.7, 1.0]	[4, 8, 16, 32, 64, 128, 256]	1080 x 1920

Table 4.14: Comparison of results of the new baseline and the new baseline with p6 and p7 removed.

	AP	**AP(50)**	**AP-s**	**AP-m**	**AP-l**
Experiment 1	**22.009**	48.585	17.267	49.365	22.207
Experiment 2	20.542	47.586	16.95	43.894	35.364

4.2.7 Relaxation of IoU Thresholds

An experiment was conducted for the new baseline where the threshold of the IoU during training was reduced, see Table 4.15. By reducing the IoU threshold in training, a larger amount of the anchor boxes will be labeled as matches. The result of lowering the IoU threshold parameters from $[0.4, 0.5]$ to $[0.3, 0.4]$ proved to be beneficial since the AP-score in all categories increased, thus it was elected to reduce the IoU threshold during training, see Table 4.16.

Table 4.15: Comparison of the new baseline and the new baseline with the IoU thresholds reduced.

	Frozen until	**IoU**	**Feature levels**	**Aspect ratios**	**Anchors**	**Input size**
Experiment 1	Res3	[0.4, 0.5]	p3-p7	[0.4, 0.7, 1.0]	[4, 8, 16, 32, 64, 128, 256]	1080 x 1920
Experiment 2	Res3	**[0.3, 0.4]**	p3-p5	[0.4, 0.7, 1.0]	[4, 8, 16, 32, 64, 128, 256]	1080 x 1920

Table 4.16: Results of the new baseline and the new baseline with the IoU thresholds reduced.

	AP	**AP(50)**	**AP-s**	**AP-m**	**AP-l**
Experiment 1	22.009	48.585	17.267	49.365	22.207
Experiment 2	**22.540**	50.321	18.223	49.881	31.199

4.2.8 Tuning of Inference Parameters

Experiments with tuning of inference parameters were conducted for three versions of the algorithm, the new baseline, the new baseline with reduced IoU-threshold, and the new baseline with p6 and p7 removed, see Table 4.17 - 4.19. The highest AP-score achieved was for the new baseline with reduced IoU threshold and thus this network was selected as the final RetinaNet, see Table 4.20.

Table 4.17: Comparision of results after tuning the score threshold.

Score threshold	**New baseline**	**Reduced IoU**	**Without P6 and P7**
0.05	22.009	22.540	20.542
0.01	**22.059**	**22.576**	**20.593**
0.20	21.571	22.410	20.287
0.30	20.788	22.297	19.783

Table 4.18: Comparison of results after tuning the NMS parameter.

NMS	**New baseline**	**Reduced IoU**	**Without P6 and P7**
0.5	22.009	22.540	20.542
0.3	21.994	22.702	20.574
0.2	22.021	**22.723**	20.576
0.1	**22.048**	22.715	**20.582**

Table 4.19: Comparison of results after tuning the topK parameter.

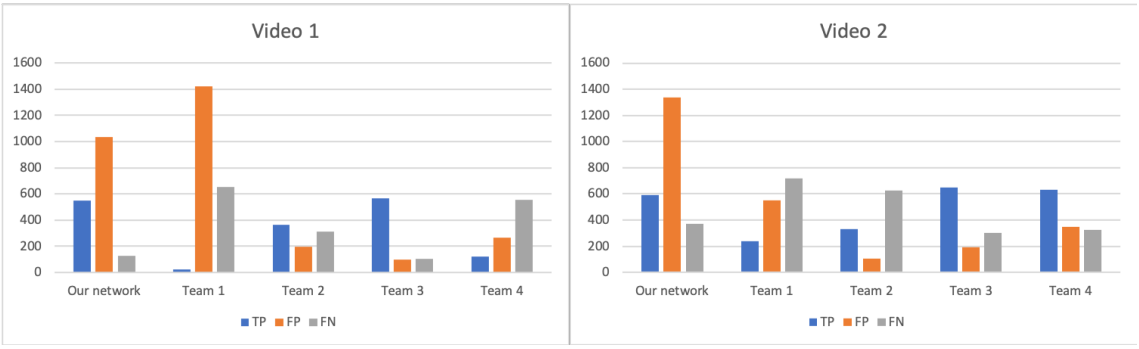
TopK	**New baseline**	**Reduced IoU**	**Without P6 and P7**
1000	**22.009**	**22.540**	**20.542**
10	22.003	22.456	20.531
2	20.936	21.161	19.936

Table 4.20: A comparison of the highest AP-scores achieved for the three versions of the new baseline after tuning the inference parameters.

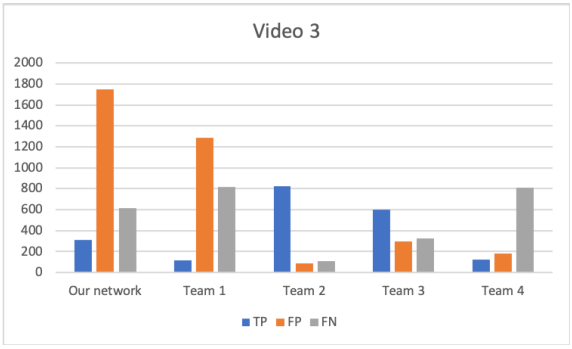
	New baseline	**Reduced IoU**	**Without P6 and P7**
AP	22.089	**22.704**	20.566

4.2.9 Results Drone vs Bird Detection Challenge

The results from the final RetinaNet, with regards to the true positive, false positive, and false negative, was compared with the contestants from the Drone vs Bird Detection Challenge, see Figure 4.1. The final RetinaNet was evaluated with the test dataset from the Drone vs Bird Detection Challenge. Compared to the other contestants in the Drone vs Bird Detection Challenge, the final RetinaNet produces a large number of false positives. However, the number of false negatives is rather low and the number of true positives is high.



(a) Comparison of results from video 1. **(b)** Comparison results from video 2.



(c) Comparison of results video 3.

Figure 4.1: Comparison of true positives, false positives and false negatives with the final RetinaNet and the networks presented in the Drone vs Bird Detection Challenge.

The winner was determined with the highest total F1 score. Team 3 was the winner however the proposed algorithm in this thesis outperforms Team 1, see Table 4.2.9. For the visual results of the algorithm, see appendix A.

Table 4.21: Comparison of F1-scores from the Drone vs Bird Detection Challenge.

	The final RetinaNet	**Team 1**	**Team 2**	**Team 3**	**Team4**
F1 score	0.35	0.12	0.68	0.73	0.41

4.3 Results Temporal Information

This section presents the results of the concatenation of feature maps, the Siamese network with addition merge, and the Siamese network with concatenation merge. The datasets used in this section are the Drone vs Bird Detection Challenge training dataset and the Drone vs Bird Detection Challenge test dataset.

4.3.1 Concatenation of Feature Maps

The implementation of the concatenation of feature maps was expensive in terms of GPU memory. Therefore, the input images were required to be cropped. The crop was performed with the center of the annotated bounding box always present in the cropped image. The crop size was defined as $[0.4 \times height, 0.4 \times width]$. The network was trained with the two different time steps and the inference parameters were tuned accordingly to Section 4.2.8. Based on the results, it was elected to not utilize this implementation of temporal information, see Table 4.22. It is clear from the results that larger time steps are more beneficial compared to smaller time steps since the AP-score increased. Both timesteps resulted in a decreased performance for small objects, a similar performance on medium sized objects, and an increased performance on large objects compared to the final RetinaNet.

Table 4.22: Results of concatenation of feature maps for the two time steps compared with the final RetinaNet.

	t-1	**t-4**	**The final RetinaNet**
AP	19.542	21.166	22.735
AP-s	15.324	16.687	18.223
AP-m	45.901	46.841	49.881
AP-l	47.206	52.346	31.199

4.3.2 Siamese Networks with Addition Merge

The Siamese networks were trained with a crop size of $[0.7 \times height, 0.7 \times width]$. The crop was performed as in Section 4.3.1 i.e with the center of the annotation always present in the crop. The network was trained for the two time steps, and the inference parameters were tuned accordingly to Section 4.2.8. Based on these results, it was elected to not use this implementation of temporal information, see Table 4.23.

Table 4.23: Results of Siamese networks with addition merge for the two time steps compared to the final RetinaNet.

	t-1	**t-4**	**The final RetinaNet**
AP	15.584	7.120	22.735
AP-s	14.511	8.322	18.223
AP-m	28.520	10.698	49.881
AP-l	16.028	15.123	31.199

4.3.3 Siamese Networks with Concatenation Merge

The siamese networks with concatenation merge were trained with a crop size of $[0.7 \times height, 0.7 \times width]$, the crop was performed as in section 4.3.1. The networks were trained with two different time steps and the inference parameters were tuned accordingly to Section 4.2.8. The concatenation merge was performed at two different locations in the network. The merge was performed after Res3 and after Res4. Based on the results of these implementations it was elected to not use these implementations of temporal information, see Table 4.24 and Table 4.25. When freezing until $Res3$ it is shown that timestep $t-4$ is performing better than timestep $t-1$. When freezing until $Res4$ it is shown that timestep $t-1$ is performing better than timestep $t-4$.

Table 4.24: Results of concatenation merge for the two time steps at Res3.

	t-1	**t-4**	**The final RetinaNet**
AP	17.382	18.804	22.735
AP-s	11.993	12.728	18.233
AP-m	41.294	47.952	49.881
AP-l	35.768	44.330	31.199

Table 4.25: Results of concatenation merge for the two time steps at Res4.

	t-1	**t-4**	**The final RetinaNet**
AP	20.904	19.368	22.735
AP-s	18.218	17.389	18.223
AP-m	38.084	35.008	49.881
AP-l	50.522	25.311	31.199

4.4 Results Semi-Supervised Learning Framework

This section presents the results of the semi-supervised learning framework. Section 4.4.1 presents the results of the visual inspection and Section 4.4.2 displays the results of the framework with regards to the Drone vs Bird Detection Challenge test dataset.
The semi-supervised learning framework was trained on the training dataset from the Drone vs Bird Detection Challenge, and with two new datasets with data collected by Saab. The first of the new datasets contained a background scene with no drones. The second contained easy examples of a drone flying in the background scene. The first of the new datasets will be referred to as Saab-train-BG, the second will be referred to as Saab-train. The semi-supervised learning framework was evaluated with the test dataset from the Drone vs Bird Detection Challenge and a new test dataset containing the drone flying in the background scene, however, with more emphasis on drones flying in front of buildings and close to treetops. The new test dataset will be referred to as Saab-test, see Table 4.26.

Table 4.26: Amount of available data for the three datasets.

	Amount of data (Frames)
Saab-train-BG	840
Saab-train	2123
Saab-test	1921

The final RetinaNet was in addition to the training dataset from the Drone vs Bird Detection Challenge, trained with the Saab-train-BG dataset. The network was utilized to generate annotations from the Saab-train data. To filter out predictions with high uncertainty, a score threshold of 0.2 was utilized. This choice of score threshold was deemed to be a good compromise between generating predictions of drones while filtering out predictions with high uncertainty. The topK and NMS inference parameters were not changed. Thereafter, the final RetinaNet was trained with the training dataset from the Drone vs Bird Detection Challenge, the Saab-train-BG dataset, and the drone data with annotations generated from the Saab-train dataset.

4.4.1 Visual Inspection of the Semi-Supervised Learning Framework

The final RetinaNet, trained with the Drone vs Bird Detection Challenge training dataset and the Saab-train-BG dataset was able to generate 1060 annotations from the Saab-train dataset and 769 annotations from the Saab-test dataset. Thereafter, the final RetinaNet was trained with the 1060 generated annotations from the Saab-train dataset and the Drone vs Bird Detection Challenge training dataset and the Saab-train-BG dataset. This version of the final RetinaNet managed to generate 1860 annotations from the Saab-train dataset, which was an increase of 800 annotations. Furthermore, this version of the final RetinaNet was able to generate 1089 annotations from the Saab-test data, an increase of 320 frames, see Table 4.27. The final RetinaNet trained with the training dataset from the Drone vs Bird Detection Challenge and the Saab-train-BG dataset will be referred to as the first network and the final RetinaNet trained with the training dataset from the Drone vs Bird Detection Challenge, the Saab-train-BG dataset and the Saab-train dataset with the 1060 generated annotations will be referred to as the second network.

Table 4.27: Comparison of the amount of data the two networks can annotate.

	The first network	**The second network**
Saab-train	1060	**1860**
Saab-test	769	**1089**

The vast majority of the generated annotations were deemed to be comparable to data annotated by a human. To further analyze the capabilities of the framework, the confidence score of annotations between the first and second networks was compared, see Figure 4.2.

(a) Prediction with the first network. **(b)** Prediction with the second network.

Figure 4.2: Comparison of predictions from the Saab-test dataset on the same image by the two networks.

The confidence scores of the second network were higher compared to the confidence scores of the first network. Furthermore, the bounding boxes matched the drone slightly more accurately. Therefore the network was deemed to be capable of utilizing data with annotations generated by the semi-supervised learning framework.

Misclassifications for the first and second network was investigated. The results indicated that with the addition of data with generated annotations, the network was able to gain more information of drone features to localize drones better. Furthermore, the network is able to reduce the number of birds misclassified as drones with additional data labeled with generated annotations, see Figure 4.3.

(a) The first network misclassified the bird.

(b) The first network does not manage to find the drone.

(c) The second network does not misclassify the bird.

(d) The second network managed to find the drone.

Figure 4.3: Comparison of misclassifications between the first and second network on the same image from the Saab-test dataset.

An additional experiment was conducted for cases when the first network and the second network misclassified a bird as a drone. Since the framework used the prediction with the highest confidence score, it is possible to accidentally annotate a misclassified bird as a drone. The experiment was performed on 24 frames where a bird was misclassified as a drone while a drone was present in each frame, see Figure 4.4.

(a) Prediction with the first network.

(b) Prediction with the second network.

Figure 4.4: Comparison of misclassifications of the first and second network.

The experiment was based on the amount of birds in the 24 frames that were falsely annotated as drones. The result of this experiment was then compared for the two

networks, see Table 4.28.

Table 4.28: Comparison of amount of falsely annotated birds.

	Number of drones found	**Number of wrong annotations**
The first network	19	24
The second network	23	17

The results of this experiment show that the first network manages to find 17 out of 24 drones, but annotated 24 out of 24 birds as drones. The second network, however, managed to find 23 out of 24 drones and annotated 20 out of 24 birds as drones. This indicates that even in difficult cases where the bird is misclassified as a drone the majority of the time, the addition of the data with generated annotations manages to reduce the number of misclassifications.

4.4.2 Semi-Supervised Learning Framework with Regards to the Drone vs Bird Detection Challenge

The first and second network was evaluated with the F1-score on the Drone vs Bird Detection Challenge test dataset, see Table 4.29. For video 3 the performance of the algorithm continued to increase when data with generated annotations from the semi-supervised learning framework was added. However, for videos 1 and 2 the best result was achieved with only the background data.

Table 4.29: F1-score on the annotated Drone vs Bird Detection Challenge test data with the inclusion of generated annotations and background.

	The final RetinaNet	**The first network (Background)**	**The second network (Background + generated annotations)**
Video 1	0.48	**0.72**	0.53
Video 2	0.40	**0.64**	0.60
Video 3	0.20	0.25	**0.34**
Total F1-score	0.35	**0.49**	0.48

Since the F1-score did increase for video 3, the semi-supervised learning framework was deemed to be beneficial since it was able to increase the performance of at least one of the test videos from the Drone vs Bird Detection Challenge test data.

5
Discussion

This chapter will discuss in-depth the results achieved. The discussion of the implemented RetinaNet is presented in Section 5.1. The result of the Drone vs Bird Detection Challenge is discussed in Section 5.2. Section 5.3 discusses the temporal implementations and Section 5.4 discusses the Semi-supervised learning framework. Lastly, suggestions for future work are presented in Section 5.5.

5.1 Discussion of Results of RetinaNet

Freezing until Res3 resulted in the highest AP-score. This indicated that the pre-trained parameters could be utilized as feature extractors for the general features of the drones. When freezing at higher layers, the pre-trained parameters failed to extract specific drone features. When freezing at lower layers, the network suffered from overfitting since the distribution of the training dataset deviated too much from the test dataset in the Drone vs Bird Detection Challenge.

The inclusion of p2 proved to be beneficial for detecting smaller objects since lower feature levels have feature maps with higher resolution and thus are suitable for detecting smaller objects. However, the computational expense for including p2 was high, since higher resolution feature maps require more computations.

The original implementation of the RetinaNet elected to place anchors of different sizes at each feature level. By performing a study of the receptive field on each feature level, appropriate anchor sizes are selected for each level. In this work, it was elected to place anchors of the same size on all feature levels, since no study of the receptive field was conducted. There is nothing inherently wrong by placing anchors of the same size across multiple feature levels, however, this strategy introduces a few aspects to be taken into considerations. Firstly, since anchors of the same shape and size were placed across multiple feature levels, the same object may be detected across different feature levels. Secondly, there is the question of computational expense. Since more anchors require more computation, the networks demand more resources.

Higher feature levels are more suited to detect larger objects since a feature map

with lower resolution has increased semantic strength due to being processed by more convolutional layers and the increased size of the receptive field. However, the increase in semantic strength comes with a cost of loss in spatial information. The increase in the AP-score for larger objects when p6 and p7 were removed might be the result of larger objects being detected sufficient enough on the other levels in the feature pyramid. Therefore, the addition of higher levels might increase the number of multiple detections for larger objects across multiple feature levels since the same anchors are placed on each feature level. Hence it is a possibility that the larger objects are detected sufficiently enough on lower levels and the inclusion of higher feature levels only results in multiple detections.

A lowering of the IoU threshold leads to an increase in accuracy since lower quality matches that could contribute meaningful information will be included. Another benefit of having more anchors labeled as foreground is the reduction of class imbalance. However, the downside is that if the threshold is set too low, the lower quality matches might instead provide false information to the network and thus reduce accuracy since the number of false positives increases. Furthermore, since the same anchors are placed on multiple feature levels, a decrease of this threshold might increase the probability that the network detects the same object on multiple feature levels.

The experiments with the inference parameters indicated that it was beneficial for the final RetinaNet to include many uncertain predictions to find additional matches. In conjunction with electing to use the same anchors across all levels, this would lead to an increase in false positives. Therefore, lowering the NMS threshold proves to be beneficial since this parameter filter out a portion of detections across multiple levels.

5.2 Discussion of Result Drone vs Bird Detection Challenge

The final RetinaNet is, compared to the other contestants in the Drone vs Bird Detection Challenge, quite aggressive and produces a large number of false positives. However, the final RetinaNet rarely fails to classify drones, and this results in a large number of correct classifications. The behavior of the algorithm could be explained by the relatively low IoU threshold during training and the anchor placement. Due to the anchor placement, anchors of the same shape and size are placed across multiple feature levels. This might result in smaller objects being detected on feature levels more suitable for detecting larger objects during training. The decrease of the IoU-threshold further increases the probability that these matches occur since more anchors, in general, will be labeled as matches. The low resolution of higher feature levels dilutes the detectable features and the distinct drone features become more general for small objects at higher feature levels. The diluted features result in

a network capable to detect relatively specific features for drones. However, the features are not specific enough since similar features are found in other objects and thus these objects are classified as drones. This will increase the number of false positives since features that share some similarities to the diluted drone features will be regarded as drones. Due to the same reasoning, the network manages to find the more difficult matches since the network is trained to classify features only similar to drones as drones, and thus, the number of positive matches increases. Overall, this aggressive approach leads to a result that is adequately comparable to the state of the art algorithms since the increase in detections partially outweighs the increase in misclassifications. There are benefits for not failing to classify these specific kinds of objects since drones might be weaponized. Therefore, as a design choice, it could be elected to rather find these drones at the cost of more false predictions, than the other way around.

5.3 Discussion of Results of Temporal Information

This Section will discuss the achieved results for the different implementations of temporal information. Furthermore, a discussion regarding the utilization of temporal information will be included.

5.3.1 Concatenation of Feature Maps

The implementation of this version of temporal information required large computational resources in terms of memory. This was expected since two separate sets of parameters for $Res4$ and $Res5$ were required to be saved since these two layers were not frozen. Furthermore, two sets of feature maps from all the feature levels also need to be stored in the memory since the concatenation was performed after the FPN backbone had extracted these features. Due to this, the input image needed to be cropped to fit inside the memory of the GPU.
An increase in performance for a larger timestep could be explained by the content of the data. The majority of data contained drones from far away and therefore the motion of the drone between frames is far less noticeable with only $t-1$ compared to $t-4$. Since the movement of objects closer to the camera is more noticeable between frames, the frame difference is more significant for these objects. Therefore, the network was capable of extracting this information and thus the accuracy for larger objects was increased. On the contrary, the movement of smaller objects between frames is far less significant. Therefore, the frame difference will not contribute with useful information. This might result in an abundance of useless information that the classifier can not find any useful patterns in. Thus the accuracy was reduced when utilizing time step t-1 compared with t-4.

The limited GPU might have affected the result since the images required to be cropped to fit inside the GPU. The images were cropped with the center of the drones always present in the image. In this case, the position of the drone in the

Drone vs Bird Detection Challenge training dataset is mainly up in the air. Inferring elements such as building and vegetation will not be included in the training process if the image is cropped to aggressively. When these elements are introduced during testing, the network has only been trained with images of drones and their immediate background. Therefore a large number of false positives may occur once objects such as buildings and vegetation are present. The semi-supervised framework proved to reduce the number of false positives significantly when only utilizing additional background. This further indicates the requirement of diverse background samples during training. Hence to obtain a better result, cropping without always centering the object might be beneficial.

5.3.2 Siamese Networks with Addition Merge

Using Siamese networks to integrate temporal information is far less computationally expensive compared to the method of concatenating feature maps, therefore more contextual information will be provided due to the ability to use a larger input size.

This method, however, proved to be unsuccessful. Instead of utilizing the average between the feature maps, the network was handed a feature map where the information had been corrupted. Compared to the previous method, an increase in the timestep decreased the performance of the network. The network might be unable to integrate the average of the feature maps and thus, a more diverse pair of feature maps will corrupt the resulting feature map further. Since the difference between feature maps is more noticeable between larger timesteps, the network partially managed to suppress the influence of the second frame at timestep $t-1$. However, as the timestep increased, the change was more noticeable and the network failed to suppress this change.

5.3.3 Siamese Networks with Concatenation

The result of this experiment showed that merging through concatenation is more beneficial than merging through addition. When merging after $Res3$, it was once again shown that the detection of larger objects was improved through the use of temporal information since the information between frames is significant enough for the network to be able to utilize it. On the contrary, the difference between smaller objects is not significant enough and thus only provide useless information. Furthermore, a larger timestep is proved to be more beneficial since this will further increase the difference between frames and thus make it easier for the network to utilize the temporal information.

A smaller time step is more beneficial when merging after $Res4$. This might be because when merging after $Res4$, only $Res5$ is available to process the temporal information. When merging after $Res3$, there were two layers available for handling temporal information and extracting features. When only a single level is required

to handle temporal information by itself while simultaneously extracting features, the single layer is not sufficient enough to handle these two tasks. Therefore, it elects to focus on feature extraction since the pre-trained parameters are tuned for that task. Since *Res*5 now focuses on feature extraction, the temporal information is only corrupting the feature map and thus reduces accuracy.

5.3.4 Utilizing Temporal Information

The methods for temporal information proposed in this thesis failed to produce a higher accuracy compared to the final RetinaNet. The highest accuracy when integrating temporal information is achieved when the merge is performed higher up in the network. This indicates that the network utilizes the full capacity of *Res*4 and *Res*5 to extract features.

When concatenating the feature maps before the classification head, all of the features have been fully processed by the ResNet backbone. However, the small classification head is unable to fully utilize the temporal information.
When merging at *Res*3 the network is able to recognize the temporal information since it has *Res*4, *Res*5 and the classifier head to process it. However, this comes at the cost of not utilizing the feature extraction fully which results in overall reduced accuracy.

In the work by D. Iglesia et al. [5], a frame difference channel is included at the bottom of the network. This method proves to be successful since the temporal information proves to increase accuracy. Based on these results and the results in this thesis, a larger part of the network might be required to be trainable, since the network need to be able to extract all the necessary features as well as handling temporal information.

5.4 Discussion of the Semi-supervised Learning Framework

This Chapter will discuss the results from the visual inspection of the semi-supervised learning framework and the results achieved on the Drone vs Bird Detection Challenge test dataset.

5.4.1 Visual Inspection

The visual inspection made on the generated annotations showed that a vast majority of the generated annotations were similar to the work done by a human annotator. It was noted that not only did the confidence score of predictions on drones increase with the addition of data with generated annotations, but the annotations seemed to fit the drone more accurately as well. This indicates that both the classification

and the localization capabilities of the network improved with additional data. Even though the generated annotations contain noise in the form of misclassifications and bounding boxes with a slight off-set to the actual drone, the benefits from the accurate predictions outweigh the downsides. One can, therefore, draw the conclusion that the network was able to at least partially utilize the annotations to iteratively improve the quality of the annotations in the next cycle. The generated annotations are deemed to have the potential to be comparable to hand annotated data.

The number of detections on the training and test datasets increased between iterations. This was an indication that the network managed to utilize the generated annotations since not only did the visual quality of the annotations increase, the number of drones that were detected also increased.

The final evaluation was based on how misclassified examples changed between iterations. These results showed that with the inclusion of more data annotated with the framework, the number of misclassifications decreased. In hard cases when both the first and second network detects a bird as a drone, the second network had improved since it was able to detect more drones and being more confident in drone detection than bird detection in some cases. This shows that with more data with generated annotations, more drone features had been learned by the network and thus it was able to somewhat reduce the number of falsely detected birds that would have been annotated as drones. Furthermore, the second network showed capabilities of being able to entirely remove false detections of birds while simultaneously localize the actual drone in cases where the first network only detected the bird.

5.4.2 Semi-Supervised Learning Framework with Regards to the Drone vs Bird Detection Challenge

The increase in F1-score for training the network with additional background data could be explained by the nature of the background data. Since the network was trained with data with a richer distribution, the network receives a larger amount of information in the form of negative examples. It was hypothesized that by including more negative examples, the number of false positives could be decreased. In the work by [49], negative examples are incorporated in the task of detecting ships from satellite images while using the fast-RCNN architecture. Similarly to the case in this thesis, a lot of false positives would occur and to reduce the number of false positives, negative examples were included in the training data much in the same way as the proposed method in this thesis. The results showed that by incorporating negative examples, the number of false positives could be reduced drastically.

When adding the data with generated annotations and evaluating on the Drone vs Bird Detection Challenge test data, the results showed an increase in performance on video 3 and a reduction in performance on video 1 and 2. This indicates that in some cases, the generated annotations can be used to improve the performance of the algorithm. Video 3 is visually close to the domain of the data with generated annotations with regards to the background and drone positions, however,

videos 1 and 2 are not. There is, unfortunately, no quantitative way to evaluate the performance of the semi-supervised learning framework in the domain of the Saab data, due to the lack of hand annotated data. However, based on the results of the visual inspection and the improvement on a hand annotated example close to the domain, this indicates that the semi-supervised learning framework can be utilized as an annotation-tool.

5.5 Future Work

Since the approach with exploiting temporal information did not improve the results it would be interesting to experiment further with transfer learning and perhaps try to freeze until $Res1$ and perform the merge thereafter. It would also be interesting to try the temporal implementations without cropping the images. One could also experiment with larger time steps to further investigate the impact of the time step. Another approach could be to concatenate or merge more time steps to incorporate even more temporal information to the network.
More investigations regarding the semi-supervised learning framework should be carried out to obtain more than a rough performance of the framework. It would be interesting to evaluate the semi-supervised learning framework on a hand-annotated dataset in the same domain as the unannotated drone data collected for this thesis and utilize the AP-metrics. Since the number of false positives decreased when background data was added it would be interesting to train the network with more specific background data on objects that proved to be hard to classify, for example birds. Another approach could be to investigate if the framework is adaptable to other objects than drones and if it could handle the classification of multiple object classes. To further test the performance of the framework it would be interesting to implement another type of deep learning algorithm, for example a two stage detector, and evaluate the semi-supervised learning framework.

6
Conclusion

In this thesis, one final RetinaNet optimized for small drone detection is presented. The final optimal RetinaNet utilizes transfer learning and anchors of the same size are placed on all feature levels. The IoU threshold during training is reduced and the aspect ratio is adapted to the specific data used in the Drone vs Bird Detection Challenge. It was discovered that the proposed methods for integrating temporal information were not advantageous due to the network's inability to both extract features and take advantage of the temporal information. It was concluded that the incorporation of data with generated annotations and background data increased the performance of the algorithm. In particular, the inclusion of unannotated background data proved to be beneficial for reducing false positives. The suggested approach of placing the same anchors across all feature levels resulted in a fairly aggressive RetinaNet. This approach resulted in a large number of true positives and false positives. The inclusion of unannotated background data heavily reduced the number of false positives while simultaneously keeping a high number of true positives. This resulted in a large increase in performance. The aggressive approach might be beneficial in real-life applications since background data are easily obtainable and require no manual labor in the form of hand-annotation. The overall purpose of the thesis was to detect and classify drones and not classify birds as drones. The results show that this object detection task can be carried out with the proposed approach, however, it could not outperform state-of-the-art supervised learning results on the Drone vs Bird Detection Challenge. The resulting framework did however show remarkable promise of being able to be used as an annotation tool for unannotated data, however, a more quantitative evaluation of the semi-supervised learning framework is required.

Bibliography

[1] A. Coluccia, et al. "Drone-vs-Bird Detection Challenge at IEEE AVSS2019," 2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), Taipei, Taiwan, 2019. [Online] Available: https://ieeexplore.ieee.org/document/8909876, Accessed on: Jan 30, 2020

[2] M. Nalamati, A. Kapoor, M. Saqib, N. Sharma, M. Blumenstein. "Drone Detection in Long-range Surveillance Videos", in: 2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS). [Online] Available: https://ieeexplore.ieee.org/document/8909830, Accessed on: Jan 30, 2020

[3] V. Magoulianitis, D. Ataloglou, A. Dimou, D. Zarpalas, P. Daras. "Does Deep Super-Resolution Enhance UAV Detection?", in: 2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS). [Online] Available: https://ieeexplore.ieee.org/document/8909830, Accessed on: Jan 30, 2020

[4] C. Craye, S. Ardjoune. "Spatio-temporal Semantic Segmentation for Drone Detection", in: 2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS). [Online] Available: https://ieeexplore.ieee.org/document/8909830, Accessed on: Jan 30, 2020

[5] D. Iglesia, M. Mendez, R. Dosil, I. Gonzales."Drone detection CNN for close and long range surveillance in mobile applications".

[6] C. Fu, M. Shvets, A. Berg. "RetinaMask: Learning to predict masks improves state-of-the-art single-shot detection for free". [Online] Available: https://arxiv.org/pdf/1901.03353.pdf, Accessed on: Jan 30, 2020

[7] T.Lin, P. Goyal, R. Girshick, K.He, P.Dollar. "Focal Loss for Dense Object Detection". [Online] Available: https://arxiv.org/pdf/1708.02002.pdf, Accessed on: Jan 25, 2020

[8] G. Sistu1, S. Chennupati, S. Yogamani. "Multi-stream CNN based Video Semantic Segmentationfor Automated Driving" [Online] Available: https://www.scitepress.org/Papers/2019/72484/pdf/index.html, Accessed on: Jan 30, 2020

[9] D. Chanyati, A. Arymurthy. "Multiple human tracking using retinanet features siamese neural network and hungarian algorithm".[Online] Avaliable: https://www.slideshare.net/iaeme/multiple-human-tracking-using-retinanet-features-siamese-neural-network-and-hungarian-algorithm, Accessed on: Mar 10, 2020

[10] X. Wang. "Human Detection in a Sequence of Thermal Images using Deep Learning". [Online] Available:

https://library.itc.utwente.nl/papers_2019/msc/gfm/WangXinran.pdf, Accessed on: Mar 10, 2020

[11] E. Sangineto, M. Nabi, D. Culibrk, N. Sebe. "Self Paced Deep Learning for Weakly Supervised Object Detection" [Online] Available: https://arxiv.org/pdf/1605.07651.pdf, Accessed on: Mar 20, 2020

[12] Colaboratory Google. "Frequently Asked Questions" [Online] Available: https://research.google.com/colaboratory/faq.html, Accessed on: Mar 24, 2020

[13] "Convolutional Neural Networks for Visual Recognition" [Online] Available: https://cs231n.github.io/neural-networks-1/, Accessed on: Mar 01, 2020

[14] A. Sharma. "Understanding Activation Functions in Neural Networks" [Online] Available: https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0, Accessed on: Mar 02, 2020

[15] P. Jain. "Complete Guide of Activation Functions" [Online] Available: https://towardsdatascience.com/complete-guide-of-activation-functions-34076e95d044, Accessed on: Mar 02, 2020

[16] R. Gomez. "Understanding Categorical Cross-Entropy Loss, Binary Cross-Entropy Loss, Softmax Loss, Logistic Loss, Focal Loss and all those confusing names." [Online] Available: https://gombru.github.io/2018/05/23/cross_entropy_loss/, Accessed on: Mar 03, 2020

[17] C. Bourez "About loss functions, regularization and joint losses : multinomial logistic, cross entropy, square errors, euclidian, hinge, Crammer and Singer, one versus all, squared hinge, absolute value, infogain, L1 / L2 - Frobenius / L2,1 norms, connectionist temporal classification loss" [Online] Available: http://christopher5106.github.io/deep/learning/2016/09/16/about-loss-functions-multinomial-logistic-logarithm-cross-entropy-square-errors-euclidian-absolute-frobenius-hinge.html, Accessed on: Mar 01, 2020

[18] Z. Feng, J. Kittler, M. Awais, P. Huber X. Wu. " Wing Loss for Robust Facial Landmark Localisation with Convolutional Neural Networks." [Online] Available: https://www.researchgate.net/publication/321180616_Wing_Loss_for_Robust_Facial_Landr Accessed on: Mar 01, 2020

[19] C. Fu, M. Shvets, A. Berg. "RetinaMask: Learning to predict masks improves state-of-the-art single-shot detection for free" [Online] Available: https://arxiv.org/pdf/1901.03353.pdf, Accessed on: Feb 01, 2020

[20] J. McGonagle, G. Shaikouski, C. Williams. "Backpropagation" [Online] Available: https://brilliant.org/wiki/backpropagation/, Accessed on: Feb 04, 2020

[21] F. Shahbaz. "Best Optimization Gradient Descent Algorithm" [Online] Available: https://medium.com/@faisalshahbaz/best-optimization-gradient-descent-algorithm-4ca5a3be3776, Accessed on: Feb 10, 2020.

[22] C. Nicholson. "A Beginner's Guide to Backpropagation in Neural Networks". [Online] Available: https://pathmind.com/wiki/backpropagation, Accessed on: Feb 3, 2020

[23] S. Saha. "A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way". Towards Data Science. [Online] Available:

https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53, Accessed on: Feb 1, 2020

[24] D. Sarkar. "A Comprehensive Hands-on Guide to Transfer Learning with Real-World Applications in Deep Learning" [Online] Available: https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a, Accessed on: Feb 1, 2020

[25] R. Asokan. "Neural Networks Intuitions: 5. Anchors and Object Detection" [Online] Available: https://towardsdatascience.com/neural-networks-intuitions-5-anchors-and-object-detection-fc9b12120830, Accessed on: Feb 10, 2020

[26] P. Ganesh. "Object Detection : Simplified". Towards Data Science. [Online] Available: https://towardsdatascience.com/object-detection-simplified-e07aa3830954, Accessed on: Feb 1, 2020

[27] Pyimagesearch, "Intersection over Union (IoU) for object detection", 2016. [Electronic image]. Available: https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

[28] T.Lin *et al.*, "Focal Loss for Dense Object Detection", Facebook AI Research (FAIR), 2018. [Online]. Available: https://arxiv.org/pdf/1708.02002.pdf, Accessed on: 2020-02-19

[29] "Anchor Boxes". Dive into deep learning. [Online]. Available: https://d2l.ai/chapter_computer-vision/anchor.html, Accessed on: Feb 7, 2020

[30] K. Sambasivarao. "Non-maximum Suppression (NMS)" [Online]. Available: https://towardsdatascience.com/non-maximum-suppression-nms-93ce178e177c, Accessed on: Feb 7, 2020

[31] S. Tsang. "Review: ResNet — Winner of ILSVRC 2015 (Image Classification, Localization, Detection)" [Online]. Available: https://towardsdatascience.com/review-resnet-winner-of-ilsvrc-2015-image-classification-localization-detection-e39402bfa5d8, Accessed on: Jan 27, 2020

[32] J. Brownlee. "How to Fix the Vanishing Gradients Problem Using the ReLU". [Online]. Available: https://machinelearningmastery.com/how-to-fix-vanishing-gradients-using-the-rectified-linear-activation-function/, Accessed on: Jan 30, 2020

[33] K. He, X.Zhang, S.Ren, J. Sun. "Deep Residual Learning for Image Recognition" [Online]. Available: https://arxiv.org/pdf/1512.03385.pdf, Accessed on: Feb 2, 2020

[34] S. Kostadinov, "Understanding Backpropagation Algorithm". [Online]. Available: https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd, Accessed on: Feb 2, 2020

[35] Towards Data Science, "a residual block", 2017. [Electronic image]. Available: https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035, Accessed on: Jan 27, 2020

[36] Medium, "ResNet Architectures", 2018. [Electronic image]. Available: https://medium.com/@14prakash/understanding-and-implementing-architectures-of-resnet-and-resnext-for-state-of-the-art-image-cf51669e1624, Accessed on: Feb 13, 2020

[37] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S.Reed, C.Fu, A. Berg. "SSD: Single Shot MultiBox Detector" [Online]. Available: https://arxiv.org/pdf/1512.02325.pdf, Accessed on: Feb 17, 2020

[38] L. Weng. "Object Detection Part 4: Fast Detection Models" [Online]. Available: https://lilianweng.github.io/lil-log/2018/12/27/object-detection-part-4.html, Accessed on: Feb 16, 2020

[39] J,Hui, "Understanding Feature Pyramid Networks for object detection (FPN)", Medium, Mar. 2018. [Online]. Available: https://medium.com/@jonathan_hui/understanding-feature-pyramid-networks-for-object-detection-fpn-45b227b9106c, Accessed on: Jan 27, 2020

[40] T. Lin, P. Dollar, R. Girshick, K. He1, B.Hariharan, S. Belongie. "Feature Pyramid Networks for Object Detection" [Online]. Available: https://arxiv.org/pdf/1612.03144.pdf, Accessed on: Jan 28, 2020

[41] Towards Data Science, "RetinaNet", 2018. [Electronic image]. Available: https://medium.com/@14prakash/the-intuition-behind-retinanet-eb636755607d, Accessed on: Jan 25, 2020

[42] J,Hui, "mAP (mean Average Precision) for Object Detection", Medium, Mar. 2018. [Online]. Available: https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173, Accessed on: Jan 28, 2020

[43] C, Nicholson, "Evaluation Metrics for Machine Learning - Accuracy, Precision, Recall, and F1 Defined", Pathmind, [Online]. Available: https://pathmind.com/wiki/accuracy-precision-recall-f1, Accessed on: Jan 25, 2020

[44] T. Arlen. "Understanding the mAP Evaluation Metric for Object Detection" [Online]. Available:https://medium.com/@timothycarlen/understanding-the-map-evaluation-metric-for-object-detection-a07fe6962cf3, Accessed on: Jan 26, 2020

[45] D. Soni. "Supervised vs. Unsupervised Learning" [Online]. Available:https://towardsdatascience.com/supervised-vs-unsupervised-learning-14f68e32ea8d, Accessed on: May 26, 2020

[46] D. Lee. "Pseudo-Label : The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks" [Online]. Available: http://deeplearning.net/wp-content/uploads/2013/03/pseudo_label_final.pdf, Accessed on: May 20, 2020

[47] N.Zeng. "RetinaNet Explained and Demystified" [Online]. Available: https://blog.zenggyu.com/en/post/2018-12-05/retinanet-explained-and-demystified/, Accessed on: Feb 20, 2020

[48] COCO, Common Objects in Context. [Online]. Available: http://cocodataset.org/#detection-eval, Accessed on: Feb 20, 2020

[49] L. Gao, Y. He, X.Sun, X.Jia, B.Zhang. "Incorporating Negative Sample Training for Ship Detection Based on Deep Learning" [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6387301/, Accessed on: Feb 27, 2020

A

Appendix 1

Figure A.1: Visual results from video 1. The drone is detected, however a false positive also occur

Figure A.2: Closer inspection of the detected drone from video 1

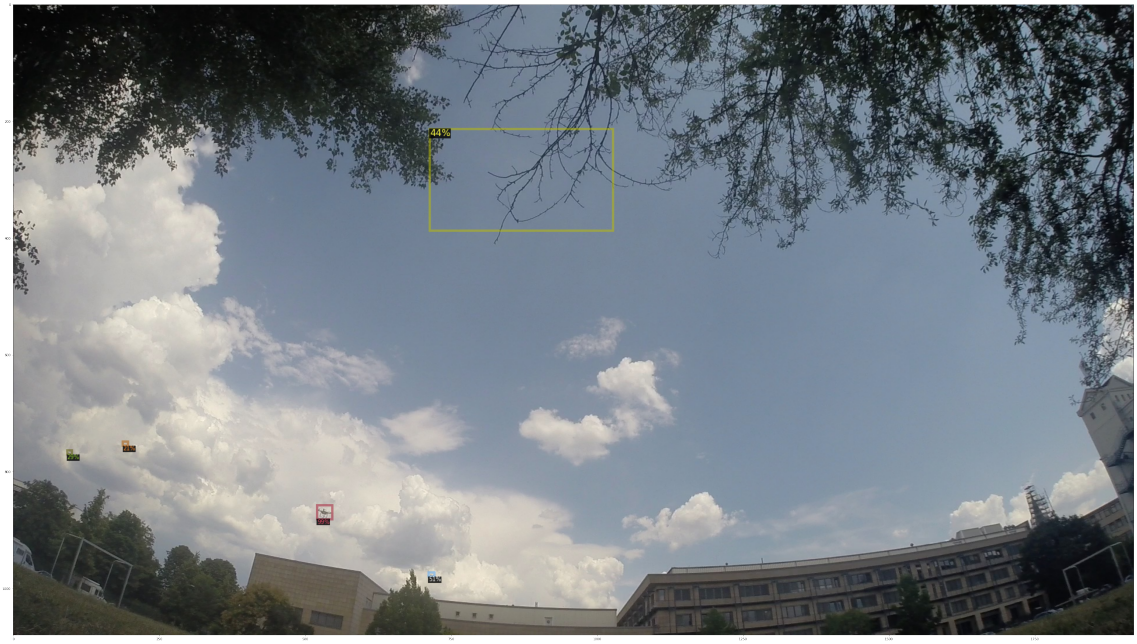


Figure A.3: Visual results from video 2. The two drones are detected, however three false positive also occur

Figure A.4: Closer inspection of the detected drones from video 2

Figure A.5: Visual results from video 3. The drone is detected and the nearby bird is not classified as a drone. However there is also a false positive

Figure A.6: Closer inspection of the detected drone from video 3