Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Jonas Hermansen Muribø

Locating Sheep with YOLOv3

Master's thesis in Informatics
Supervisor: Professor Svein-Olaf Hvasshovd
June 2019

NTNU
Norwegian University of
Science and Technology

Jonas Hermansen Muribø

Locating Sheep with YOLOv3

Master's thesis in Informatics
Supervisor: Professor Svein-Olaf Hvasshovd
June 2019

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Summary

Recent years have seen an increase in the use of unmanned aerial vehicles in numerous fields, ranging from power line inspections to wildlife monitoring. One of these fields is locating free-ranging sheep, specifically at the end of the grazing period for the roundup.

This thesis examines how well YOLOv3 (You Only Look Once), a one-stage object detector, is able to locate sheep in drone footage. Additional objectives were to examine how some modifications to the network affected performance. The first modification was how the network would make detections; either attempt to detect sheep as a superclass, or detect white, black and brown sheep separately as subclasses. The other modification was to a series of network parameters that related to input resolution, and confidence threshold for predictions. YOLOv3 was implemented through a popular fork of Darknet, an open-source framework for neural networks.

The most important metric was to find as many sheep as possible, and this was achieved by detecting sheep as a superclass, with a resolution of 832x832 pixels and a confidence threshold of 0.1. 12 out of 1650 sheep in the test set were not detected, resulting in a recall of 99%.

There is, however, some uncertainty to how applicable the network is; all footage was of sheep on pasture with hardly any change in lighting, and 84% of all sheep in the dataset were white. Additionally, the low threshold caused an overabundance of bounding boxes for some predictions. Further work should attempt to capture substantially more data for training to increase generalisability.

Sammendrag

De siste årene har ubemannede flyvende fartøy hatt en stor vekst innen mange forskjellige fagområder, fra inspeksjoner på strømnettet til overvåking av vilt. Et av disse feltene innebærer å lokalisere sau på beite, spesifikt i forbindelse med innsankning nær slutten av beiteperioden.

Hovedfokuset i denne oppgaven er å undersøke i hvilken grad YOLOv3 (You One Look Once), en enkelt-stegs objektgjennfinnings-algoritme, er i stand til å lokalisere sau i drone-bilder. Ytterligere mål involverer å undersøke hvordan noen endringer i network påvirker ytelsen. Den første endringen består i å endre hvordan nettverket klassifiserer et funn; nettverket kan enten forsøke å lokalisere og detektere sau som en kategori, eller prøve å separere forskjellig fargede sau som svarte, hvite eller brune. Resterende endringer består av å endre oppløsning, og variere terskelen for hvor sikker nettverket skal være i et funn for at det skal klassifiseres som en sau. YOLOv3 ble implementert gjennom en populær variant av Darknet, et rammeverk for nevrale nettverk med åpen kildekode.

Å finne så mange sau som mulig var regnet som det viktigste målet, og dette ble oppnåd ved å gjennfinne sau som en overordnet kategori, med en oppløsning på 832x832 piksler, og en terskel på 0.1. 12 av 1650 sau i testsettet ble ikke funnet, noe som resulterte i en gjennfinningsrate på over 99%.

Det er dog noe usikkert i hvor anvendeling nettverket er på dette stadiet; alle bilder er av sau på inngjerdet beite med liten varians i lys og datasettet består av rundt 84% hvite sau. I tillegg har den lave terskelen for gjennfinning ført til usikkerhet knyttet til antall funn per sau. Videre arbeid burde forsøke å anvende betraktelig mer og varierende data for å øke anvendeligheten.

Preface

This thesis was carried out as a Master's thesis as the culmination of the Master's of Science in Informatics programme at the Norwegian University of Science and Technology, Trondheim in 2018/2019.

I would like to thank Steingrim Horvli for allowing me to capture footage of his sheep, and Professor Svein-Olaf Hvasshovd for supervising all work related to this thesis. Your input has been invaluable.

Table of Contents

Abbreviations

AI	=	Artificial Intelligence
ANN	=	Artificial Neural Network
CNN	=	Convolutional Neural Network
DNN	=	Deep Neural Network
DP	=	Deep Learning
FLIR	=	Forward-Looking InfraRed
FOV	=	Field Of View
GCS	=	Ground Control Station
GPS	=	Global Positioning System
NN	=	Neural Network
R-CNN	=	Region Convolutional Neural Network
RNN	=	Recurrent Neural Network
UAV	=	Unmanned Aerial Vehicle
YOLO	=	You Only Look Once

Chapter 1

Introduction

Each year approximately 2.1 million sheep range freely in Norway which in itself presents a lot of challenges [39, 20]. One challenge is locating smaller groups of sheep that have strayed from the flock at the end of the grazing period. The process of manually searching and traversing a large area is both tedious and time-consuming.

A popular solution to this problem is to radio tag a portion of the free-ranging sheep to aid in recovery. This method drastically reduces the time spent herding sheep at the expense of costly equipment. The tags are typically around 1000 NOK a piece (not including required subscriptions) [10, 32, 36, 31], making it expensive to tag every sheep. Luckily, sheep tend to stick together in flocks and ewes with their lambs, and therefore tagging only a portion of the sheep is often sufficient to recover the majority.

Fall roundups in Norway usually consist of three steps: two major roundups by use of radio tags where more than 90% of sheep are gathered, and a final roundup to locate all of the stragglers [15]. While the two first roundups are quite efficient, the last one is the most troublesome; there could be no information on the whereabouts of the last sheep and the farmers are by law required to roundup all of the surviving sheep and document their losses.

Recent years have seen enormous growth in applications involving unmanned aerial vehicles (UAV), resulting in more affordable commercial and industrial products. Internationally, UAVs are used in a wide range of applications, and some farmers in Norway have already started utilising UAV's for oversight during the grazing period, and have reported great success in doing so [13]. Manually sifting through all of the footage and marking positions of sheep, however, is a time consuming and error-prone process that could benefit from automation, especially in regard to large areas.

One of the state-of-the-art object detection and localisation solutions is You Only Look Once (YOLO), and more specifically YOLOv3. YOLOv3 has a $mAP@50$ (mean average precision with 50% IoU (Intersection over Union) as the threshold) of 57.9 with an

inference time of $51\,\mathrm{ms}$, and while it is not the most accurate it is likely one of the best trade-offs between precision and speed currently available [28].

Currently, a large portion of the roundup is spent trying to locate a small portion of sheep. A lot of resources are invested in the sheep, and the farmers are by law required to account for all sheep that have been released in the grazing period, which means that it is not feasible for them to simply stop looking for the sheep despite how long it might take. By using a drone with automated sheep detection software, it would be easy to examine large areas in a short amount of time to swiftly locate nearby sheep.

This thesis revolves around examining possible software solutions to handling the automated detection aspect of a tool to aid farmers in roundup at the end of the grazing period.

Chapter 2

Literature Review

This chapter gives an overview of existing research related to localisation of livestock, game, planes, and cars from UAV footage, as well as ways to analyse this footage efficiently. The purpose of this chapter is to give an overview of relevant research and state-of-the-art techniques for object detection in images.

Developing a fully automated system for recognising objects, animals and people have a multitude of uses in modern society. New advances in deep learning (DL) are making these techniques prevalent in the field of object detection. As will be discussed below, some notable applications include wildlife monitoring and research, as well as civil engineering. There are numerous possible military applications as well, however, due to several challenges this field takes on a rather slow stance on integrating deep learning into their systems.

2.1 Earlier Master's Thesis

The problem of locating sheep in drone imagery was also a topic in a thesis from 2018, however, with somewhat different scope. The thesis was divided into two parts. The first part consisted of planning a flight path for optimal area coverage, and the second part was a comparison between a feature engineered computer vision technique and two different deep neural networks. The dataset used for comparison consisted entirely of thermal images captured from a fixed-wing drone, and due to some issues with interference during flight, most images contained notable noise.

The thesis concludes that a classical computer vision approach performs best overall as compared to RetinaNet [18] and their implementation of a convolutional neural network (CNN). The authors argue that this is likely due to the small dataset used for training as

deep neural networks often require large amounts of data to outperform traditional techniques. The results indicate that using deep learning and thermal images is a highly relevant solution to object detection.

2.2 Wildlife

Conservation workers, farmers, and researchers are often groups with limited budgets, and drones have repeatedly proven to be both a cheap and efficient solution to aerial monitoring. The combination of automated flights with object localisation has further improved this process, as demonstrated in a paper by Van Gemert et al. from 2015 [38].

Van Gemert et al. performs experiments with a quadcopter carrying a GoPro Hero 3 Black across fields with cattle. The experiments aim to evaluate the effectiveness of a fully automatic system to aid conservation workers with examining population trends and identifying threats. While their dataset was not recorded in the wild, the use of the quadcopter allowed for recording at a variety of heights, speeds, and camera angles, and it is argued that the recording setup closely matches a real-world setup. The experiments showed the most promising results with a technique called Exemplar SVM (Support Vector Machine) with a precision and recall of 0.66 and 0.72 respectively at a rate of 0.9 sec/image.

A similar set of experiments were performed by Rik Smit in 2016 [33]; a commercial drone was flown over fields with cattle, followed by a comparison of different machine learning algorithms on the output data post-flight. Smits results top out at 0.446 and 0.742 for precision and recall; while the recall is similar, the precision reported by Van Gemert et al. is significantly better.

Both of these studies demonstrate the viability of using some kind of object localisation on UAV footage, however, they are not ready to replace manually annotated data due to a lack in both recall and position. They can, on the other hand, be used as an aid in analysing the footage. It is worth to note that all recordings were taken in open fields with some man-made structures present which might not represent the performance experienced in real-world applications.

2.3 Civil Engineering Applications

A paper by Radovic et al. [23] demonstrates success in using a modified, YOLO-inspired [26] CNN to locate airplanes in aerial images taken by UAVs. The paper proposes an architecture with 24 convolutional layers followed by two fully connected layers as shown in Figure 2.1.

Radovic et al. were able to achieve an object classification accuracy of 97.8% when tested on aerial images at varying levels of resolution. It is, however, worth to note that no average Intersection over Union (IoU) was specified in the paper, i.e. how accurate the

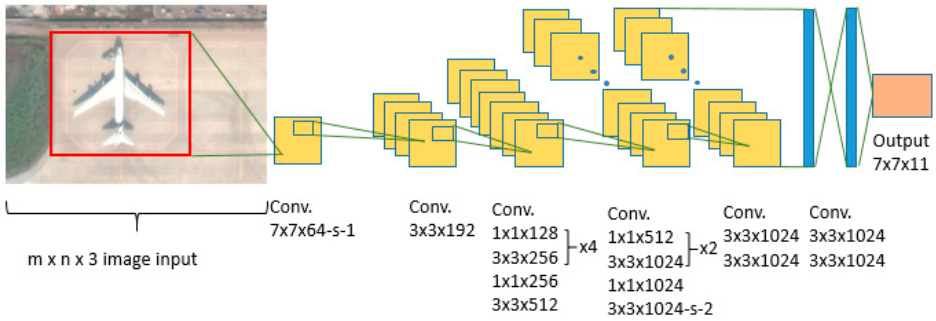


Figure 2.1: Illustration of CNN architecture (from Radovic et al., 2017).

predicted bounding box of the object was, and as such it is difficult to judge the overall quality of their predictions, but the results are promising.

Benjdira et al., [4], perform a comparison between YOLOv3 and Faster R-CNN (Region-CNN), two state-of-the-art CNN algorithms, on UAV footage of cars at different altitudes in an urban environment. They conclude that YOLOv3 is on par with or outperforms Faster R-CNN on all of the evaluation metrics, and especially the processing time of $0.057\,\mathrm{s}$, compared to $1.39\,\mathrm{s}$ for Faster R-CNN, makes YOLOv3 very attractive for real-time operations.

These papers by Benjdira et al., and Radovic et al. presumably lack in generalisability as both datasets are generated from similar environments; daytime lighting in urban areas. While airplanes are less likely to be present in rural areas, cars are ubiquitous in modern society and it remains to be seen whether the trained networks perform as well on a wider variety of environments. Both papers indicate that YOLO, or its derivatives, may be efficiently applied to object localisation on drone imagery.

Other civil engineering uses include, but are not limited to: construction site monitoring, solar farms inspection, traffic monitoring, oil, gas, and power line inspections.

2.4 Military Applications

While UAVs have been utilised by intelligence and military agencies for years, there is a reluctance to apply deep neural networks (DNN) to localisation systems. A paper by Svenmarck et al. from 2018 [35] summarise the reasoning for this in three main categories:

- Lack of transparency and interpretability - It is difficult to legitimise a decision made by a DNN due to its black box nature. The output from the network may have drastic consequences, and it is challenging to understand the reason behind the output; in some cases, the output could be caused by broader statistical correlations as opposed to the desired deciding factors.

- Vulnerable to adversarial attacks - By manipulating the input signal of a DNN it is possible to drastically change the output, even if the model is unknown. This has been demonstrated by Su et al. [34] by changing as little as one pixel to entirely change the outcome of the DNN. Su et al. demonstrate a success rate of up to 72.85% on a basic DNN, which could potentially be disastrous.
- Data - There is a concern about whether or not the data previously gathered can be used successfully for machine learning. It might be challenging to adjust data pipelines to adapt to new data requirements.

None of these categories apply to the problem at hand; Firstly, there is no specific need to legitimise the object detections as they are intended as an additional aid, and not a replacement. Secondly, there is not expected to be a need to secure the network against attacks. Lastly, there is no current data pipeline that can be utilised for this problem.

Military applications, or limits thereof, are mainly mentioned here due to the obvious advantages for tactical applications and the like, and this section aims to illustrate why most of the literature research is based on civilian applications.

2.5 Summarised

Several surveillance and monitoring papers, as well as the earlier Master's thesis, utilise a thermal imagery system rather than a visible imagery one [12, 5, 6], and it is not hard to see why. Thermal cameras can detect subtle differences in temperature, for instance between a warm bodied animal and its background, therefore making it quite suitable for discovery-style applications. Thermal cameras on drones are not considered standard equipment, especially on off-the-shelf drones, and may as such require substantial investment. Ultimately, this means that footage is to be captured by the use of an easily available and cheap off-the-shelf drone.

Several different deep neural network architectures have been tested, as well as traditional computer vision techniques, however, YOLO has demonstrated superior performance. Although the results are promising, it is worth to note that the subject of study has mostly been mechanical, e.g. cars and planes, and the environment urban. It is unclear whether YOLO will perform at the same level on sheep in rural areas,

Chapter 3

Theory

This chapter will briefly explain convolutional neural networks, its constituents and how they learn. Finally, it will explain the main aspects of the single-stage object detector YOLOv3, You Only Look Once v3.

3.1 Artificial Neural Networks

In the modern sense of the word, artificial neural networks (ANNs) are networks of interconnected nodes inspired by the neurons in the brain. The nodes are typically grouped together in layers, and a series of layers make an ANN. While not a new invention, the use of ANNs has grown immensely in recent years and has seen applications in finance, marketing, production, monitoring, and more [21].

YOLOv3 is a variation of a convolutional neural network (CNN), which in turn is a variation of an ANN. To explain the relevant fundamentals of YOLOv3 it is easiest to start at the bottom by explaining convolution.

3.2 Convolution

In image processing, convolution is an operation where a matrix K, often called kernel, is applied to another matrix I. The operation itself is called convolving I by K. Usually denoted by $*$, convolution is quite similar to a dot product of K and I, see Figure 3.1. K is fitted into the upper left corner of I so that there is full overlap and each weight in K is multiplied with the corresponding overlapping value in I and then summarised resulting in $I * K$. The kernel then slides across the entire image with a given stride; in this example, the stride is set to one. Note that a convolution normally decreases each dimension of I

by the corresponding dimension in K minus one; this is also demonstrated in Figure 3.1. This is because the result of the convolution is placed at the centre of the kernel, and in this instance, I goes from 7x7 to 5x5 due to K being 3x3, i.e. a reduction of two in each dimension. This can be avoided by padding I with a one-pixel wide border of zeroes and is useful if there might be important information along the edges of the input or if it is desired to maintain the size of the input.

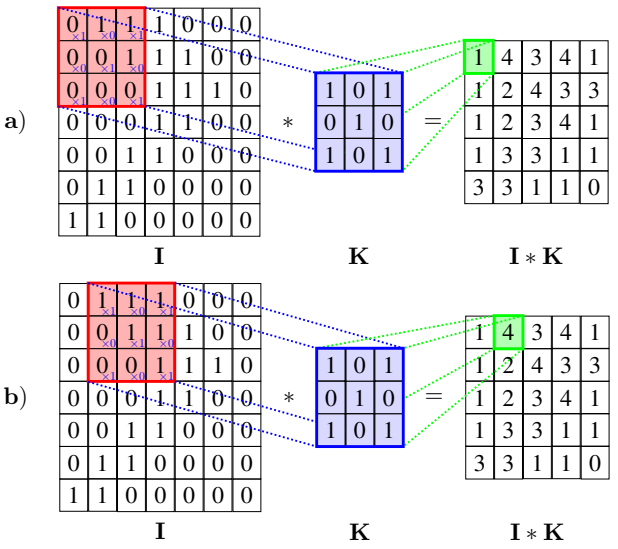


Figure 3.1: Convolving image I with the kernel K ($I * K$). The first step is shown in ***a*** where the kernel is placed in the upper left corner, and the next step shown in ***b*** where the kernel slides with a stride of one. The process is repeated until the kernel has been slid across the entire image (images derived from [1]).

The reason why convolution is used in some ANNs is that the operation resembles the way neurons in the brain respond to visual stimuli, and as ANNs are inspired by neurons it is natural to attempt to apply techniques for visual processing as well. From a more practical perspective; convolution is a good way of detecting features and patterns in images in an efficient manner. Edges, for instance, consist of transitions between intensities or colours, and are often good descriptors of high-level information in images.

3.3 Layers

While convolution may be useful on its own, the real advantage comes from grouping convolution operations together in layers. There is a multitude of different layers, however, only the relevant ones will be explained here. Note that all examples are taken from the default *yolov3.cfg* [25], and an overview of all layers can be found in Appendix B.

3.3.1 Convolutional Layer

A convolutional layer is a layer that applies convolution with 1 kernel per input channel and one or more kernels together make up one filter. Convolution is a good way of examining the relationship between neighbouring pixels and discovering important features in images, and CNN's utilise convolution through convolutional layers.

A filter evaluates how well the input resembles a feature, e.g. edges, roundness, colour or higher level information that may be difficult for humans to understand. The same filter can be applied to all parts of the image as a feature is expected to be equally important at all locations. Multiple filters may be used, and the number of filters used directly correspond to the amount of channels output. Listing 3.1 shows an example configuration of a convolutional layer:

Listing 3.1: Convolutional Layer

```
[convolutional]
batch_normalize=1
filters=1024
size=3
stride=1
pad=1
```

which results in a layer as illustrated in Figure 3.2. Batch normalisation is explained further in section 3.6, however, its overall purpose is to improve training. The *filters*, *size*, *stride*, and *pad* parameters define the format of filters as well as how they are applied. Filters are 3x3x1024 with a stride of 1 and the input is padded with 1 so that the output retains the same dimensions as the input, see section 3.2 for an explanation of stride and pad. Note that the depth of the output correlates to the number of filters, and the depth of the input correlates with the number of kernels in each filter.

3.3.2 Shortcut Layer

A shortcut layer is a variation of a skip connection, i.e. a connection that can skip past layers. A shortcut layer concatenates the outputs of two layers: the previous one, and a second defined by the current layer minus *from* number of layers. Listing 3.2 shows a shortcut layer that concatenates the previous layer and the layer 3 before the current.

Listing 3.2: Shortcut Layer

```
[shortcut]
from = -3
activation = linear
```

3.3.3 Route Layer

Route layers are very similar to shortcut layers, as they can be used to skip layers. There are however two different kinds of route layers, one with a single parameter, and one with

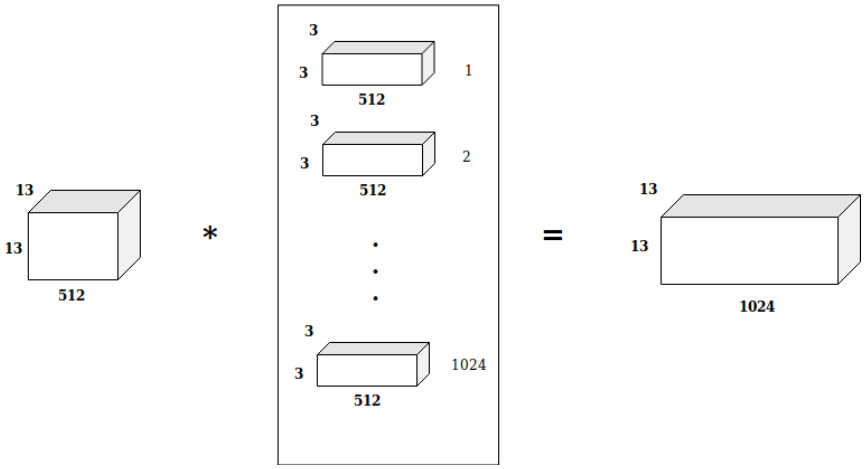


Figure 3.2: Example of a convolutional layer: an input of 13x13x512 is convolved (*) by 1024 filters of shape 3x3x512 resulting in an output of 13x13x1024. As the input was padded, the output retained the resolution of the input.

two parameters as shown in Listing 3.3.

Listing 3.3: Route Layer

```
[route]
layers = -4

[route]
layers = -1, 36
```

The first instance uses one parameter and simply outputs the layer indexed by the current layer minus 4. The second instance concatenates the previous layer, -1, and layer at index 36, however, it can be used to concatenate arbitrarily indexed layers as well.

3.3.4 Upsampling Layer

Listing 3.4: Upsampling Layer

```
[upsample]
stride = 2
```

An upsampling layer performs upsampling by a factor of *stride*, which in the instance of Listing 3.4 is 2. If the input is 13x13x256, this layer would then output 26x26x256.

3.3.5 YOLO Layer

The YOLO layer is the layer in YOLOv3 responsible for performing detection. YOLOv3 contains three of these layers that are responsible for the detection of objects at three different scales, from largest to smallest where each layer attempts to extract higher-level information than the previous.

To understand this layer, a new concept has to be introduced: **Anchors**. Anchors, or anchor boxes, are a series of predefined height-to-width ratios that are the basis for predicted bounding boxes. In a way, anchors can be considered a series of boxes that represent the most typical size and shape of objects in a dataset. Optimally, anchors are calculated with K-means clustering on the training set so that their initial shape and size is closer to the objects in the dataset. When a network makes a prediction, instead of directly outputting dimensions of the bounding box, it predicts an offset from the most similar anchor box. This technique has shown promising results in regards to improving the performance of CNN's with little overhead to implementation, training and inference time [40, 27].

Listing 3.5 shows relevant parts of the configuration for the second YOLO layer. Anchors are defined in the parameter *anchors*, and by default each layer consist of 3 out of the *num = 9* anchors defined by use of the *mask* parameter. For this layer, anchors 3, 4 and 5 are used, resulting in anchors (30, 61), (62, 45) and (59, 119) respectively as they are 0 indexed. Larger objects are detected at the first layer and smaller objects at the third. Note that the number of *classes* is set to 80 by default and does not necessarily reflect what will be used in this thesis.

Listing 3.5: YOLO Layer

```
[yolo]
mask = 3,4,5
anchors = 10,13, 16,30, 33,23, 30,61, 62,45, 59,119, ...
classes = 80
num = 9
jitter = .3
...
random = 1
```

The final parameters, *jitter* and *random*, are data augmentations the layer applies at even intervals during training. *jitter* specifies the maximum random adjustment to size and aspect ratio performed, and *random* enables resizing of the network input resolution up to a certain offset from the original resolution. Using *random* should improve network performance when using images of different resolutions as input as the network is trained with inputs of varying resolution.

3.4 Convolutional Neural Networks

A convolutional neural network (CNN) is an artificial neural network that uses convolutional layers to process data. CNN's are specifically designed to handle image input in the

shape of a multidimensional matrix and are very efficient at doing so. AlexNet, the architecture of which is shown in Figure 3.3, demonstrated great success in comparison to other computer vision techniques in the *ImageNet Large Scale Visual Recognition Challenge* in 2012 where it won the classification and localisation categories as the only competitor that used a CNN [29]. Subsequent years saw an enormous increase in the use of CNN's, and in 2014 most of the submissions for the challenge were based on CNN's.

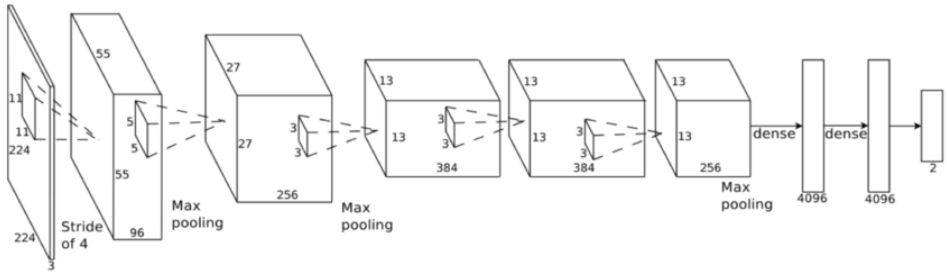


Figure 3.3: Architecture of AlexNet, a CNN from 2012(image from [11]).

Operations involving CNN's can be classified as computer vision tasks as they attempt to extract high-level information from images, and this thesis is no exception to that. Classification and object detection are two main areas of research; classification is simply to classify an image as a certain object, while object detection locates an object in the image and then classifies it. Object detection is the most challenging of these two, and also the most relevant for this thesis; images may contain zero, one or several sheep, and it is important to know the position of sheep in the image for retrieval and validation purposes. Traditionally, object detection performed by multi-stage detectors use two steps:

1. Predict or detect possible objects in the image and return bounding boxes for these
2. Classify the bounding boxes as an object or false positive

However, in recent years a couple of notable one-stage detectors have appeared: SSD [19], RetinaNet [18] and YOLO [26]. These detectors take an image as input and directly output bounding boxes with class predictions. The main arguments for using one stage detectors as opposed to multistage detectors is the ability to train and optimise an entire network in one go, whereas multistage solutions may consist of multiple networks that need to be trained and optimised separately resulting in additional complexity to the solution.

3.5 Back Propagation

Backpropagation is the process of adjusting the weights of an ANN so that it can be trained to solve a problem and is the typical process that is repeated when training a network. This is done by passing the input through the network, comparing the output to ground truth, i.e. expected output, and then calculating an error rate via a specialised loss function. This error rate is then cascaded backwards through the network to modify the weights (in

the instance of CNN's filters are the weights) so that the next time the same or a similar image is passed through the network, the prediction should be closer to the ground truth. A series of parameters define limits on how drastically the network can learn, i.e. how drastically weights may be changed at a time to attempt generalisability. The theory is that with enough small changes and tweaks to weights, a network of nodes is able to learn any arbitrary mapping of input to output.

Weights are usually not updated after only passing a single image through the network, but rather after passing a batch of images through. The size of the batch is usually constant when training a network, however, it is not standardised across AI frameworks. After one batch has been passed through, and the error rate backpropagated, it is called a single epoch. Note that the term epoch varies in definition and use, and epoch as a term in this thesis follows the given definition.

3.6 YOLO - You Only Look Once

YOLO is a one-stage object detector that performed very well at its release in 2015 and has since seen two major updates: YOLOv2 [27] and YOLOv3 [28] in 2016 and 2018 respectively. Each update has kept YOLO in the state-of-the-art tier among object detectors, and especially the latest version, YOLOv3, performs exceptionally well in comparison with competitors, as evidenced by [28].

As YOLO is a fully convolutional neural network, and a one-stage object detector, different parts of the network architecture are responsible for different parts of the object detection; most notable is the new feature extractor, Darknet-53, see Figure 3.4. The feature extractor is the backbone of YOLOv3 and is purely responsible for extracting features that directly relate to objectness from images. Darknet-53 is named after its constituent 53 convolutional layers and YOLOv3 consists of Darknet-53 plus additional object detection layers that total to 106 layers [16].

YOLOv3 uses a series of smart techniques to improve object detection, the most notable of which are described below (YOLO will be used to reference YOLOv3 from this point onward unless stated otherwise) [14, 26, 27, 28]:

Bounding box prediction across scales - YOLO predicts objects at three scales across the entire image within a grid with a stride of 32, 16 and 8 meaning that each cell size is defined as the network input resolution divided by stride. By default, each cell uses three anchors per scale to make predictions and the cell where the centre of the object is located is responsible for its detection. If the input is 608x608 the number of bounding boxes predicted would be

$$((608/32)^2 + (608/16)^2 + (608/8)^2) * 3 = 22743.$$

As it is not viable to work with this amount of bounding boxes YOLO performs pruning of bounding boxes in two steps: thresholding by objectness and non-maximum suppression. Each predicted bounding box is assigned a confidence that indicates how certain the

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 3.4: The architecture of the feature detector Darknet-53 (image from [28]).

network is in its prediction that an object is present in this bounding box; this is known as objectness. The default threshold for objectness during detection is 0.25 but can be altered at runtime. Non-maximum suppression is a way of removing highly overlapping bounding boxes by pruning away every bounding box except for the one with the highest confidence.

Batch normalisation - Normalisation is the process of converting the distribution of values to have a mean of 0 and a standard deviation of 1, resulting in values that are mostly between -1 and 1. Batch normalisation normalises the activation values from the previous layer, which regularises the effect of a varying distribution in the training data and speeds up training [22].

Configurable anchors - As previously mentioned YOLO uses 9 anchors for predictions. These anchors are calculated by performing k-means clustering on the dataset and dividing the output evenly across scales. By having anchors extracted from the current dataset YOLO increases mAP (mean average precision) and reduces training time.

High-resolution classifier - YOLO uses a high-resolution input of 608x608x3 and supports any multiple of 32 as input due to the grid structure used for detection. This can be modified both before training to improve accuracy at the cost of increased training time, and after training prior to inference to improve accuracy at the cost of inference time. Increasing the resolution requires additional memory.

Extensive Loss Function - From YOLO(v1) all variations of YOLO have used an extensive and effective loss function and the original function is shown in the paper from 2015 [26]. The loss function in YOLOv3 is composed of three loss-calculations [14, 28]:

- **Classification Loss** is the penalty for predicting the wrong class for a bounding

box and uses binary cross-entropy loss for each label, the details of which are not important to this thesis.

- **Localisation Loss** is a measure of how erroneous a predicted boundary box is, and is only applied to bounding boxes responsible for detecting an object. Both location and size are taken into account.
- **Confidence Loss** is entirely based on objectness. As explained, YOLO uses three anchors per cell per scale and objectness for a prediction is 1 if the anchor responsible has the highest IoU of all anchors at this scale. If the other anchors have an IoU with the ground truth of less than 0.5 (default threshold), then the predictions are penalised. If on the other hand, they are over 0.5, they incur no loss.

3.7 Transfer Learning

Training a network from scratch is a time-consuming process, however, there is a way to use a previously trained network for other purposes. Darknet-53 is the backbone for YOLOv3 and the variation of Darknet-53 used for by Joseph Redmon et al. [28] was trained on ImageNet [17] with over 20 000 categories. Transfer learning is a way of training where a previously trained network is used for another purpose. This is done by removing some of the detection and classification layers near the end of the network, and then training new detection layers with the new data. The theory behind this is that the feature extraction layers of a network have learnt to detect important features from images, and by extracting the layers and weights related to this feature extraction, a large portion of training has already been completed.

The idea behind Darknet-53 is exactly this; a high-performance backbone that can be used as a basis for a multitude of purposes so that new models do not have to train from scratch. The term training, or learning, in the context of this thesis, is that of applying transfer learning to the problem at hand, namely object detection of a custom dataset with sheep.

Chapter 4

Data Acquisition and Analysis

This chapter explains how the data acquisition was planned, executed and finally how data was analysed and labelled.

4.1 Planning

The timeframe for this thesis in regards to the grazing period was rather unfortunate; the work began within a month of the end of the grazing period and finished close to a month or two after the grazing period started again the next year. As there would be no other opportunities to record data the acquisition had to be planned and executed within a limited amount of time, possibly before the details surrounding the project had been decided.

4.1.1 Requirements

The premise of deep neural networks is primarily that their performance correlates directly to the quality and quantity of the data used to train them. While it is possible to train a neural network with limited amounts of data, this often results in an impact on the generalisability of the network.

Some basic requirements were defined in regards to the desired characteristics of the data:

- **A large number of pictures** - DNN's perform better when trained on more data.
- **A varying number of sheep per picture** - This should ensure that the network is not faultily trained to place x number of sheep in every picture.

- **A spectrum of different coloured sheep** - Sheep tend to have differently coloured wool, and some sheep have wool of multiple colours and shades. It is also possible that sheep may be dirty, and this change in colour should be accounted for as well.
- **As wide a range of environments as possible** - Different backgrounds and lighting result in widely different pictures.
- **Footage from different heights** - YOLO predicts a series of sizes for the bounding boxes, and having a range of sizes could increase generalisability.

Optimally, the data should fulfil all of these requirements to be as versatile and applicable as possible, however, lacks in most areas can possibly be supplemented with augmented data, e.g. changes in brightness, saturation, random scaling, and cropping.

4.1.2 Equipment

A DJI Mavic Pro [8] will be used to take pictures at 4000 * 3000 pixels at varying heights every 2 seconds. This particular drone was chosen due to being relatively cheap off-the-shelf consumer hardware, as well as being available in the early stages of the thesis and during roundup. Retailed at $749 it is a very affordable option that features 4k video, 12MP camera, live feed to a smartphone and 27 minutes of flight time.

Individual pictures were chosen over video due to the Mavics limitation regarding saving metadata from the flight. By taking pictures rather than video the Mavic Pro natively supports storing a range of metadata related to each picture most importantly, longitude, latitude, and altitude based on the built-in GPS, and relative altitude based on the pressure difference between the takeoff and current height. This meant that it was possible to generate an overlay of the flight path and the height from the drone to the ground during analysis, however, there were some problems with the height measurement, as will be discussed in section 4.3.

4.1.3 Location

All footage was planned to be taken at a farm named Horvlia in Lønset, Oppdal. The owner, Steingrim Horvli, graciously allowed us to fly our drone over his farm. Most of Horvli's sheep were either free ranging within a fenced area or had not yet been retrieved for the end of the grazing period. However, the numbers were unclear prior to the day of recording.

According to Google Maps and Kartverket, the farm is surrounded by boreal forest and terrain transitioning between farmland and highland. The environment is considered to be representative of a possible environment for free ranging.

No particular flight path was planned due to the unknown number and locations of sheep that were currently grazing prior to arriving at the farm. The flight path would thus have to be planned or improvised on location.

4.1.4 Analysis

When performing analysis on imagery taken by a drone, there are three main ways of digesting the data:

- **Post flight** - All data is saved on a portable storage device. An analysis is performed after the flight is finished, and the storage device retrieved. In the event of a crash, all data could be lost.
- **In-flight** - All data is sent to a Ground Control Station (GCN) capable of receiving and analysing data in real-time. Interference could render transmitted data useless, and no data is stored on the drone for later retrieval.
- **Hybrid** - Data is saved on a portable storage device and sent to a GCN for real-time analysis. When the flight is finished, the storage device is retrieved and the data analysed, checking for discrepancies with what was received at the GCN.

It is important to note that these three methods only apply to drones that are by themselves incapable of performing analysis in real-time, like the DJI Mavic Pro used in this project. While the DJI is capable of sending a live video feed to a smartphone that is connected to the remote, alternatively a smart-remote, this video feed is limited to 30 frames/second at 1920 by 1080 pixels. The quality of the video is naturally affected by interference, distance and blocking objects. At the time of recording, there is not expected to be a working solution for handling a video feed in real-time, and analysis will be performed post-flight in an exploratory fashion.

4.2 Acquisition

Images were captured the 2. of October 2018 between 11:00 and 13:00. Conditions were near perfect; sunny and lightly clouded throughout the session with hardly any wind. Despite reports of snow in earlier weeks, no snow appeared to remain on the day of capture.

A total of three flights and captures were executed:

- The first flight was over sparse forest, with what is assumed to be typical Norwegian flora and terrain; mountainous, rocky, mossy, and trees both spread and in clusters. Sheep were supposed to be in the area, however, it turned out that they had moved approximately 500 meters further away, and due to challenging terrain and legal compliance to stay in the line of sight to the drone, further exploration was discontinued. The first flight ended up resulting in a series of ”negative” images, i.e. images that were expected not to contain any sheep.
- The second flight was of a mix of open fields surrounded by dense forest. Most sheep appeared to spread out in groups across the open fields, though a fraction kept close to the edges.

- The third and final flight included the farm buildings and equipment as well as some sheep and cattle ranging close to the buildings. Mostly open fields with some roads, buildings and equipment present.

4.3 Analysis

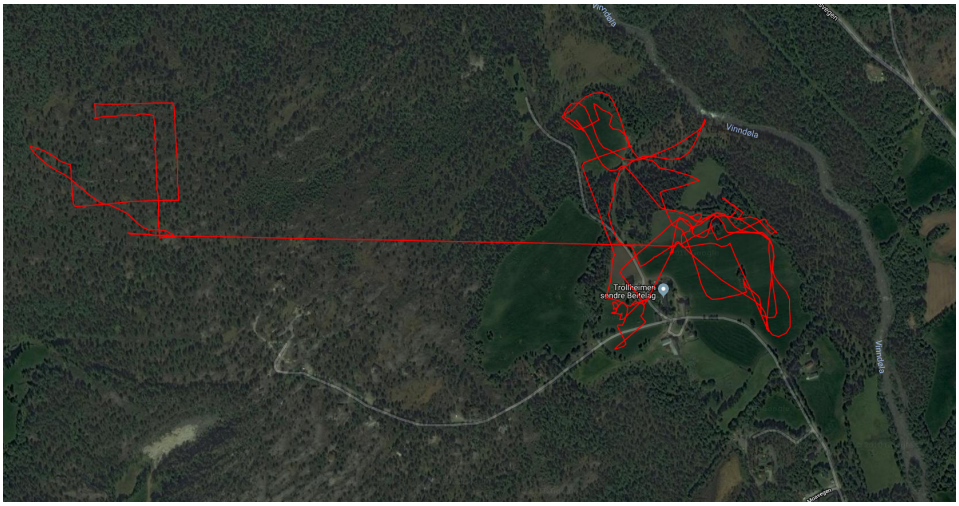


Figure 4.1: The flight path from the three flights recorded. Note that there is an interpolation between all flights, and the long line between the left and the right of the map is one of these interpolations.

A total of 942 pictures were captured from three different flights, from a varying range of heights. Some pictures were removed due to being too close to the ground or otherwise unsuitable, resulting in 844 images in the final dataset.

Due to the quite significant differences in height in the environment (90 meters difference at most), some preprocessing was required to get an estimated height difference between the drone and the ground directly beneath it. As all images were tagged with longitude and latitude at the moment of capture, it was a simple task to extract this data from the footage and draw a flight path onto a Google Maps cutout as shown in Figure 4.1.

The DJI Mavic Pro has two independent ways of measuring altitude; the first is based on the GPS and the second uses an onboard barometric pressure sensor that measures the difference in air pressure between take-off height and current height. As DJI does not specify an accuracy for either method, both will be examined here.

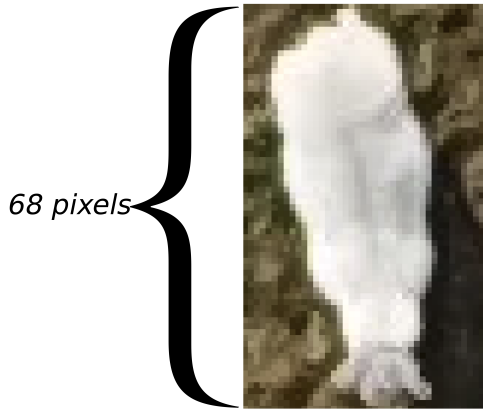


Figure 4.2: A cutout of the sheep in question from *DJI_0654.JPG*

DJI_0654.JPG, see Figure 4.2, was chosen due to a sheep standing close to parallel with the height axis of the image, and as such provides a reasonable point of reference. The calculated height based on the GPS is

$$\begin{aligned} HeightDifference &= DroneHeight_{GPS} - GroundHeight \\ &= 776m - 550m \\ &= \underline{226m} \end{aligned} \tag{4.1}$$

above the ground. This number is of particular concern as it is illegal to fly a drone above 120 meters above the ground without specific permission from 'Luftfartstilsynet' (the Civil Aviation Authority of Norway) as well as a certificate. During flight, it did not appear as though the drone was flying at this height, which raises a question about the accuracy of these measurements. The following calculations attempt to calculate the real-world size of a sheep in the image by use of the GPS altitude measurements to examine whether they are reasonably accurate.

An average adult sheep is 1.3 meters long [30], and the sheep in question was found to be 68 pixels in length. The FOV of the camera on the DJI Mavic Pro is $66.8°$ horisontally and $52.5°$ vertically according to DJI's tech support, see Appendix A. The real world vertical span, or footprint, of the image, was calculated to be

$$\begin{aligned} F_t &= 2 * h * \tan\left(FOV_t/2\right) \\ &= 2 * 226m * \tan\left(52.5°/2\right) \\ &\simeq \underline{223m}. \end{aligned} \tag{4.2}$$

From there we can calculate the approximate real-world tallness of each pixel as

$$\begin{aligned} P_t &= \tfrac{F_t}{NumberOfPixels} \\ &= \tfrac{223m}{3000px} \\ &\simeq \underline{0.074m/px}, \end{aligned} \tag{4.3}$$

resulting in a sheep that is

$$\begin{aligned} SheepLength &= PixelsTall * P_t \\ &= 68px * 0.074m/px \\ &\simeq \underline{5.0m} \end{aligned} \tag{4.4}$$

according to measurements recorded by the GPS. While the calculations are rough estimations the values are clearly incorrect, and the validity of height measurements provided by GPS is not considered to be reliable. Despite erroneous altitude reporting, the associated latitude and longitude correspond reasonably with images taken at each location, and these measurements are for all intents and purposes treated as accurate.

Calculations (4.1) through (4.4) use the altitude reported by the GPS, however, Mavic Pro has a second way of measuring altitude; the barometric pressure sensor. This sensor does not directly measure height above sea level; instead, it measures offset in height from a previous location and altitude. On takeoff, the drone automatically measures the pressure at its current height, and all further heights measured are based on an offset from this initial height. In practicality, this requires the user to know the absolute height at the takeoff position to be able to calculate anything other than the difference in height given that the sensor is accurate enough. At the image in question, *DJI_0654.JPG*, the drone recorded a relative altitude of $\simeq$ 75 meters and a ground height of $\simeq$ 549 meters. The related flight started at *DJI_0192.JPG*, whose ground height is at approximately 533 meters. This results in an absolute height of $533m + 75m = 608m$. The measured distance to the ground is as such $608m - 549m = 59m$. It follows that the size of the sheep according to the relative altitude is

$$\begin{aligned} F_t &= 2 * 59m * \tan\left(52.5°/2\right) \\ &\simeq 58m \\ P &= \tfrac{58m}{3000px} \\ &\simeq 0.02m/px \\ SheepLength &= 68px * 0.02m/px \\ &= \underline{\underline{1.36m}}. \end{aligned} \tag{4.5}$$

Compared to the average length of sheep at 1.3 meters, 1.36 seems like a reasonable estimate. The GPS relies on connections to satellites to measure its position accurately, and as the area in question is mountainous this could interfere with the signals. The barometric sensor, on the other hand, is not reliant on any external factors other than the air

pressure and, if calibrated properly, should give quite accurate results as demonstrated above.

It is, however, worth noting that these calculations do not take into consideration the natural curvature of the landscape; height differences in an image are not accounted for, and could as such lead to erroneous values. Assumptions are also made regarding the real-world size of each pixel; pixels at the centre of the image should be smaller than those at the edges due to the curvature of the lens. These calculations simply serve as a very rough estimate, and are by no means meant to be used to calculate an accurate real-world size; it does, however, give an indication of the actual size of a sheep.

As previously mentioned, it is illegal to fly above a certain height without special permissions and qualifications, and there is often a built-in security feature in consumer drones that prevent them from flying above this limit. The DJI Mavic Pro is no exception to this and enforces 120 meters above-the-ground limit at all times that intuitively has to be enforced by use of the barometric sensor, as the GPS by itself is incapable of measuring its altitude above the ground. This raises an interesting question about flight height; as the barometric sensor measures the height difference between take-off and current position. By taking off from the top of a mountain and then flying over a cliff the drone could fly at several hundred meters above the ground whilst not breaking the limitation.

The inaccuracy and limitations of the DJI Mavic Pro mean that it is not a likely candidate for any commercial use in the capacity that is desired.

4.4 Labelling

Images were labelled with **LabelImg** [37], where a rectangular bounding box was specified for each sheep discovered and its graphical interface is shown in Figure 4.3. All sheep were labelled with one of three colours; brown, black or white.

Table 4.1: Distribution of images that contain sheep.

	With sheep	Without sheep	Total
Number of images	361	483	844
Distribution	43%	57%	100%

Table 4.2: Table of sheep distribution in the entire dataset.

	White sheep	Black sheep	Brown sheep	Total
Number of sheep	4621	724	132	5477
Distribution	84.4%	13.2%	2.4%	100%

The distribution of images containing sheep is shown in Table 4.1 and the total distribution of sheep is shown in Table 4.2, and white sheep clearly constitute the largest portion of sheep in the dataset with 84.4%. An additional category to the three primary colours of

Figure 4.3: LabelImg - Tool used for labelling images

sheep was added: difficult. This tag specifies that it was difficult to label said sheep due to either multiple colours, partial obstruction or uncertainty. Note that this classification was additional, not alternative, and this distribution is shown in Table 4.3. Approximately 5.8% of all sheep were difficult to label and 22% of black sheep. Black sheep were often difficult to label due to their shadow being of a similar colour to their wool, possibly resulting in poor bounding boxes for black sheep.

Table 4.3: The distribution of difficult sheep to label across the different classes of sheep. Distribution is the portion of the relevant class that was difficult to label.

	White sheep	Black sheep	Brown sheep	Total
Number of sheep	160	159	0	319
Distribution	3.5%	22.0%	0%	5.8%

The dataset was split into training and test with a ratio of 60:40 respectively. The distribution of train:test is typically closer to 75:25, however, due to the dataset containing a large amount of similar images (as the pictures were taken within 2 seconds of one another) as well as a recommended requirement by AlexeyAB [2] of having an equal amount of images with and without sheep in the training set, the 60:40 distribution was chosen. The training set contains 500 images, 250:250, and the test set 344, 111:233.

4.5 Observations

While examining the dataset, a couple of peculiarities were discovered:

- Several pictures showed signs of loss of focus at the perimeter of the image. This was especially clear when hovering in place and then increasing altitude. This could be a limitation with the lens/camera on the DJI Mavic Pro, and there is no way to remedy this without removing large parts of the images. No illustration is provided to demonstrate this example as it was hard to examine the extent of the loss of focus on print.
- There is a surprising difference in exposure, or brightness, between some pictures taken within 2 seconds of one another. This could be due to clouds temporarily covering the sun, or it could be an issue with the camera and lens, however, it is difficult to verify this. An example is shown in Figure 4.4. From a practical perspective, it should not make much of a difference in performance as YOLO already performs some data augmentations that alter the hue and saturation of images.

Figure 4.4: Example of exposure difference between images taken within 2 seconds of each other. *DJI_0349.JPG* and *DJI_0350.JPG*

- There are only 2.4% brown sheep in the dataset and this raises the question of whether or not brown sheep are indistinguishable from black sheep in drone imagery. As the main task is to locate sheep in images and not only brown sheep there should not be an issue. This could, however, cause some issues in case farmers are interested in comparing the images to what was observed.
- Due to a mistake during acquisition, all images from the first flight were flipped 90 degrees. As this could potentially be an issue for YOLO, all of these images were flipped back 90 degrees so that all images were horisontally aligned.

Chapter 5

Experiment Structure

This chapter details the research questions, the experiments derived from these questions and the metrics used for evaluation.

5.1 Research Questions

Based on the motivation and literature review, the following research questions are proposed:

RQ1 How well does YOLOv3 locate sheep in UAV footage?

RQ2 Is it easier to classify sheep as a superclass, or as one of three subclasses based on colour?

RQ3 Is it possible to increase performance noticeably by tweaking parameters post training?

See Figure 5.1 for an overview of the relationship between the superclass and subclasses.

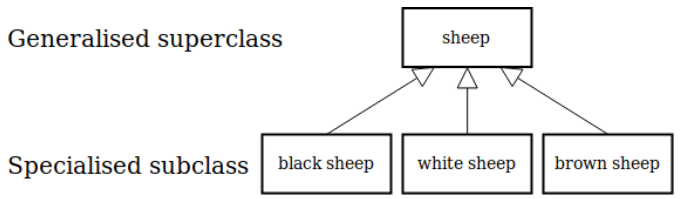


Figure 5.1: Class diagram illustrating the relationship between the superclass and subclasses.

5.2 Experiments

RQ1 is not examined by experiments on its own, but rather through RQ2 and RQ3 due to the nature of these questions as modifications of YOLOv3 parameters. Furthermore, RQ3 is applied to both the superclass and subclass configurations mentioned in RQ2, meaning that all research questions are closely tied together and are rather treated together as a concept than separately. Ultimately, all three questions are collectively answered through the experiment structure detailed below and are not strictly assigned individual experiments.

The hyperparameters related to the superclass and subclass separate the two configurations into two top-level categories. Each of these categories are subsequently examined through additional modifications to parameters that, as opposed to the hyperparameters, can be altered after network training is completed. Preliminary tests indicate that performance typically peaks and stabilises within 12000 epochs with a stable loss per epoch of < 1, and is as such the target amount of training for both superclass and subclass. Weights are saved at 1000 epoch intervals, and only the weights that result in the highest recall and precision are used for testing.

Three different resolutions for network input have been chosen. Configuring different resolutions was chosen due to two reasons, the first being a recommendation by AlexeyAB [3], the creator of the code repository used, that increasing resolution before or after training may improve performance, albeit at the cost of additional training and inference time respectively. Secondly, increasing the network input size will increase the amount of information that can be input, which could be important as sheep in the images are quite small. 608x608 pixels is the resolution used in the YOLOv3 paper and is a such a natural base resolution, and as lower resolutions are not expected to improve results, the other values are higher than 608x608 pixels. 832x832 and 1024x1024 pixels were chosen as two higher resolutions for performance evaluation. Note that all resolutions have to be a multiple of 32 due to how YOLOv3 is structured, details of which are explained in Section 3.6.

By decreasing the resolution of all images by a factor of 4, from 3000x4000 to 750x1000 pixels, training time is decreased as YOLO has to perform less scaling for images to fit within 608x608 pixels. As this scaling results in images that are smaller than the higher resolution input configurations, unscaled images are used during performance evaluation.

Additionally, each of these 6 configurations will be examined at different threshold levels, i.e. by changing the limit for how confident the network has to be per prediction for it to count as a prediction. A lower threshold will increase the recall, as even predictions the network are not very confident in, will be assumed to be correct. With a higher threshold, the network has more confidence in the predictions it makes, take for instance a threshold of 0.7; for a prediction to count, the network has to be at least 70% confident in its prediction.

Finally, two specific parameters were edited to optimise YOLO specifically for small objects. Small objects in this instance are defined as objects smaller than 16x16 pixels after

images are downsized to 608x608 pixels. As with most other parameters, these too are recommended AlexeyAB [3], and the first is anchor recalculation. As explained in section 3.6, anchors define a series of height-to-width ratios, and every bounding box prediction is defined as an offset from these anchors. By having multiple bounding boxes at different scales, YOLO performs better on overlapping objects, and thus possibly on objects that are close to each other as well. The reason for needing to recalculate these is that the default anchors are too large for what is expected to be found in the images, take for instance the last and largest anchor; At 373x326 pixels, a sheep would have to span about half the image for this ratio to be correct, while the average size is $< 10\%$, i.e. 61x61 pixels or less for most instances.

The second set of parameters consists of increasing upsampling from 2x to 4x at layer 97 as well as changing which layers are concatenated at layer 98. Why exactly these layers were recommended is not known, however, it is likely due to some attributes of small objects that are more easily detected at certain layers.

5.3 Performance

Performance data is generated by applying different configurations of YOLOv3 to the dataset of labelled sheep and then comparing the differences. The performance will be measured against ground truth, i.e. the manually annotated bounding boxes detailed in section 4.4, and will consist of 4 different metrics that showcase different advantages and disadvantages. The 4 metrics are:

- **Precision** - Ratio of correct classifications on predicted bounding boxes
- **Recall** - Ratio of predicted bounding boxes and ground truth
- **Inference time** - Time to evaluate a single image
- **mAP@50** - Mean average precision with a threshold of 0.5 intersection over union (IoU).

The best performance is achieved by the configuration that results in the highest recall. This is based on the idea that this thesis aims to evaluate a solution for a sheep locating tool, and the most important aspect of this tool is to locate every sheep. While it might be unrealistic to locate every sheep, selecting the configuration that results in the highest recall is expected to be the most useful for farmers during a roundup.

That being said, additional metrics are being used as well to demonstrate other practical aspects of the configuration, namely precision and inference time. $mAP@50$ is mainly provided to examine how well YOLOv3 performs on this dataset as compared to the *COCO test-dev* dataset [7], a commonly used dataset for performance comparison of modern object detection frameworks. As briefly mentioned in chapter 3.6, 57.9 is the documented $mAP@50$ of YOLOv3.

5.4 Hardware

A single compute node with the following specifications will be used for all neural network training and evaluation:

- **CPU** - i7-9700K 8 cores
- **GPU** - RTX 2080 Ti 11GB
- **RAM** - 16GB

Chapter 6

Results and Discussion

In this chapter, the results are presented, discussed and examined for validity.

6.1 Performance

Both superclass and subclass configurations were trained 12000 epochs (see Section 3.5 for an explanation on epochs) as performance appeared to stabilise at this point, and this was achieved after approximately 20 hours with no notable difference between the two. Each configuration was evaluated on 12 different sets of weights, from 1000 to 12000 epochs, and the weight that resulted in the highest recall and precision, where recall was prioritised, was used. The number of epochs for all configurations is shown in Table 6.1, and on average, training with subclasses achieved peak performance 1000 epochs earlier than training with a superclass. There was no instance where the 12000 epoch-weights were the highest performing weights, and the highest resolution for both subclass and superclass achieved its peak performance earlier than the other resolutions.

Figure 6.2 shows the precision/recall trade-off for varying thresholds, and each plot represents a structure and network resolution labelled accordingly. As shown, the results are quite good across the board with a precision of [0.87-0.99] and recall of [0.81-0.99] for the different configurations. Some preliminary tests were run without optimisation for small

Table 6.1: The number of epochs trained before the best performance was achieved. Maximum number of epochs was 12000.

Epochs	608x608	832x832	1024x1024	Average
Superclass	9000	9000	5000	7667
Subclasses	7000	7000	6000	6667

Table 6.2: True positives, and false positives and false negatives for the different configurations at a threshold of 0.1. The total number of sheep in the test set was 1650, the sum of true positives and false negatives. The best result for each column is highlighted.

Structure	Resolution	True Positives	False Positives	False Negatives
Superclass	608x608	1622	101	28
	832x832	**1638**	**98**	**12**
	1024x1024	1633	153	17
Subclasses	608x608	1601	152	49
	832x832	1617	145	33
	1024x1024	1602	249	48

objects, and these resulted in a maximum precision and recall of 0.92 and 0.88 respectively, which indicates that making these optimisations caused a drastic improvement in performance. At each threshold, the lowest performing superclass and resolution achieved equal or superior performance to the best configuration using subclasses. Further details on the performance of each configuration are shown in Table 6.2 and an example from the best configuration, superclass at 832x832 pixels, is shown in Figure 6.1.

Figure 6.1: An example of a near perfect prediction where each sheep in the image is assigned a single bounding box with high IoU. From *DJI_0682.JPG* with superclass at 832x832 pixels and a threshold of 0.1.

Likewise, $mAP@50$ is also quite high as shown in Table 6.3, and notably higher than the 57.9 reported by Joseph Redmon [28] by at least 33.8. As mentioned in Section 5.3, recall,

and typically precision, are the main performance metrics, and these $mAP@50$ values were not taken into account when examining what configuration performed best.

Inference times, as shown in Table 6.4, are achieved by taking the average evaluation time for each threshold [0.1-0.9] and then dividing this by the total number of images in the evaluation set. It is also possible to examine execution time for individual images, however, the chosen method more closely resembles a real-world situation where a series of images are analysed in bulk. When using higher resolution network inputs, the inference time increases slightly, which is likely due to the increased processing time for larger images as there are simply more pixels. What is interesting is that despite images with 1024x1024 pixels having close to three times the amount of pixels as images with 608x608 pixels, the increased processing time per image is only about 12-13%. The memory requirements, on the other hand, are expected to be directly correlated to the size of the input, however, memory usage was not recorded during inference.

It does, however, seem like inference time for both superclass and subclasses are practically the same, which is as expected. The image size for both configurations are the same, the structure is practically identical, except for a couple of extra filters and output size that are caused by having more classes when working with subclasses as opposed to the superclass.

Table 6.3: $mAP@50$ at different resolutions with either the superclass or subclass structure.

mAP@50	608x608	832x832	1024x1024
Superclass	98.3	99.1	98.9
Subclasses	91.7	93.2	92.8

Table 6.4: Average inference time for a single, unedited image at different resolutions with either the superclass or subclass structure.

Inference Time	608x608	832x832	1024x1024
Superclass	62.02ms	65.89ms	69.77ms
Subclasses	62.34ms	64.92ms	70.41ms

6.2 Discussion

The first and most important aspect to examine is the recall and precision of all configurations, especially the discrepancy between superclass and subclasses. When trained with sheep as a superclass, the network consistently performed better than when trained on the subclasses based on the primary colour. Take for instance the best performing configurations for superclass and subclass; the ones with a resolution of 832x832 pixels. The subclasses were able to find most sheep leaving only 33 out of 1650 sheep outside of detection, while the superclass was able to detect all but 12. Both configurations performed exceptionally well on the same data basis which indicates that hyperparameters and parameters can make a notable difference on the outcome.

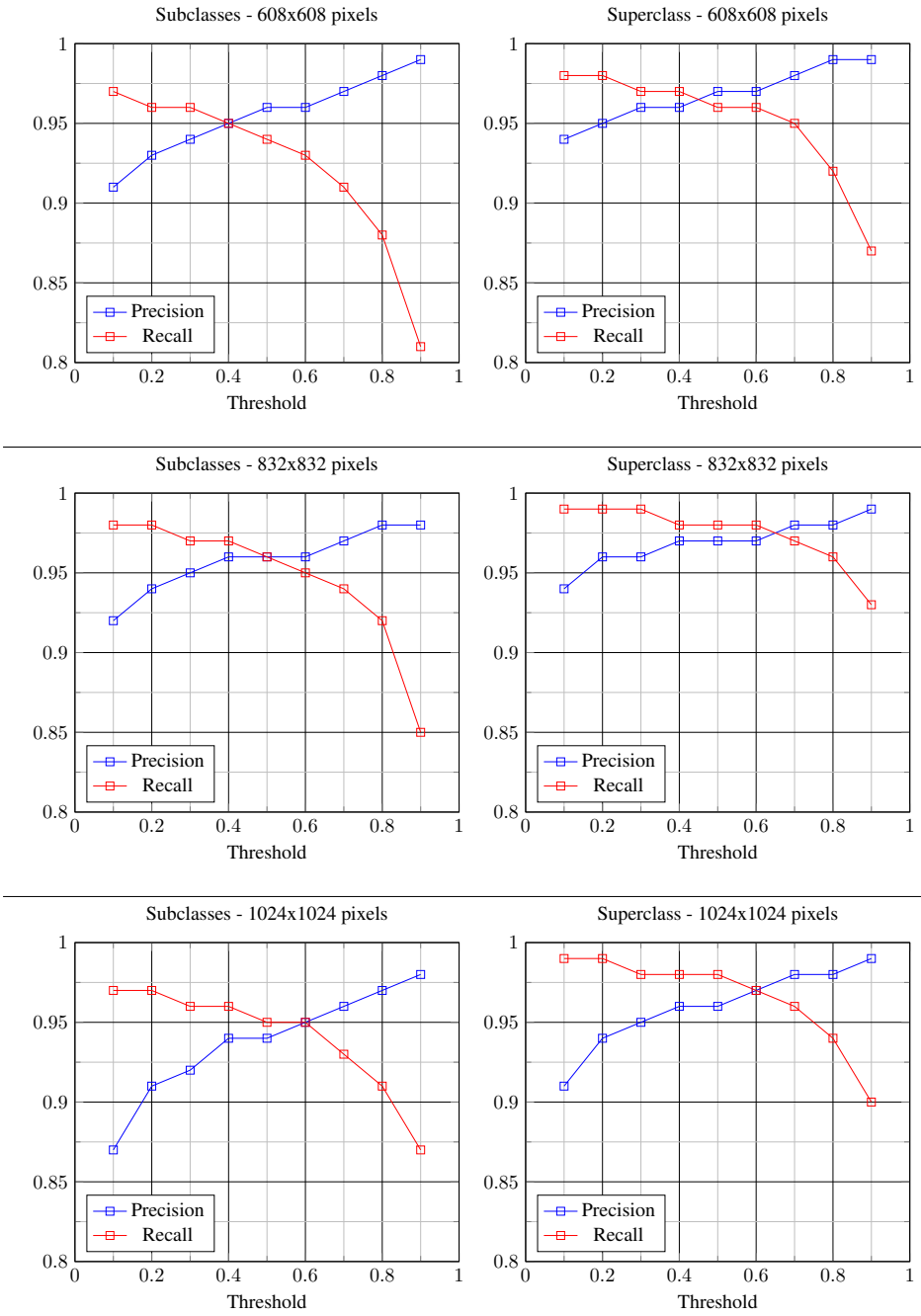


Figure 6.2: Precision and recall from inference with all configurations and varying threshold.

The expected culprit for this discrepancy is the distribution of differently coloured sheep. Assume that the only difference between sheep is the colour of their wool; if a network is able to accurately learn the shape, size, rotation or some other high-level feature related to sheep, that might be difficult for humans to understand, then it should result in a colour-insensitive network. This could potentially explain the difference in results between the superclass and subclasses; both configurations have the exact same basis, however, each subclass could be separately attempting to define what a sheep looks like, with a base assumption that it has to be of a certain colour to be classified as such. This assumption, in addition to white sheep being the main constituent in the dataset with 84%, should result in a network that is good at detecting white sheep, and worse at detecting black and brown sheep. When working with a superclass, the main difference is colour, and if there is a colour invariant pattern to sheep that can be learnt, chances are that YOLOv3 is capable of learning it. This theory is backed up by the absolute number of sheep that were not detected, 33 (2%) and 12 (0.72%) for subclasses and superclass respectively.

A second theory that slightly contradicts the one proposed above is based on how YOLOv3 learns about objects through objectness. As mentioned in 3.6, objectness is the confidence of a network in a predicted bounding box containing an object. As YOLOv3 assigns and penalises objectness independently of classification and localisation, given that the data basis is identical, both of the network configurations should be equally capable of detecting all sheep as their objectness is independent of their class. Based on this assumption, recall for both the superclass and subclasses should be practically the same.

There is an uncertainty that applies to both of these theories that have not yet been addressed; the difficulty label. While the label itself was not used during training or inference the fact that a total of 6% of all sheep were difficult to label as a colour remains. The working theory is that while these sheep were difficult to label as a single colour mostly due to being multi-coloured, the superclass configuration does not care about what colour the sheep is, but rather if a sheep is present or not. This could explain why the superclass was able to achieve slightly better results than the subclasses and that precision is noticeably better.

Moving on to $mAP@50$, the results show that even the worst $mAP@50$ achieved, 91.7, is considerably better than YOLOv3's reported 57.9 [28]. There are two main reasons for this, both being directly dependent on the number of classes. The dataset that Redmon et al. use to train and evaluate YOLOv3 contains 80 different classes, as opposed to either the one or three classes used in this thesis. These 80 classes are mostly independent of each other, e.g. *elephant*, *skateboard* and *pizza* do not necessarily have as many features in common as sheep. This variety introduces the first reason for why the presented network performs so much better than the one used by Redmon et al.; there are simply fewer features to learn and look for. When working with more classes, and especially classes that are not interchangeable if colour is discarded, it will naturally take longer to train due to the complexity separating multiple classes. The second reason is closely related to the first; with fewer classes, there is a smaller chance that YOLO will confuse classes with each other. If a situation occurs when the network is uncertain about a classification yet the objectness is high, having only three classes, as opposed to 80, to choose from increases the chance that the prediction is correct.

While investigating why the highest resolution classifiers gave negative returns, the first of two mistakes made during the thesis became clear. Originally it was assumed that the parameter *random* in the configuration file enabled upscaling of the input image of 608x608 pixels up to a certain value for every 10 iterations. What the parameter actually does is temporarily increase the network input resolution so that the input image can have higher resolution. In the case of an original resolution of 608x608 pixels, this peak higher resolution was 896x896 pixels. This would typically not be an issue as the network scales the input to fit the network resolution, however, all images were scaled down to 750x1000 pixels to increase training speed. By pure luck, this resolution is still higher than 896x896 pixels which resulted in the network being trained as intended if *random* is enabled as the fork of Darknet used for this thesis resizes the input image without keeping the aspect ratio.

In regards to the difference in performance between 832x832 and 1024x1024 pixels, it is highly likely that this is due to the network having been trained on images with resolutions between 608x608 and 896x896 pixels. Features within images of 608x608 and 832x832 pixels are of a known size as the network has been trained on similarly sized examples, however, features in images with 1024x1024 pixels are too detailed and with features too large for the network to perform optimally. From Table 6.2 it is also clear that the number of false positives increases drastically when moving to 1024x1024 pixels for both the superclass and subclasses which lends credibility to this theory.

The second mistake made was not a critical mistake either, however, it could potentially have evened out the results between superclass and subclass; hue and saturation. YOLOv3 performs a random amount of data augmentation at regular intervals, and two of these augmentations alters the colour. Saturation alters the vibrancy of colours, and hue rotates the colour palette. By having both of these enabled the span that is brown, black and white as colour ranges increase. The main issue with this is that the difference between black and brown decreases. An augmented brown sheep may appear black and vice versa, which could lower the precision of these two classes. This should, however, not affect recall in any significant manner; a black sheep being detected as a brown sheep is still a sheep located.

Finally, inference time was not on par with Joseph Redmon et al.'s reported 51 milliseconds [28], and was approximately 62 milliseconds for the corresponding resolution. As there is no practical difference in network architecture or the likes, the only plausible cause for this increase inference time should be tied to the original size of the input. When running inference, images with a resolution of 3000x4000 pixels were used and each image had to be scaled to fit within 608x608 pixels, and this operation is the expected reason behind the additional 11 milliseconds of inference time. As the COCO dataset contains a large number of images [7], it would not be viable to store or use high-resolution images for object detection because it would take up an enormous amount of space. Additionally, most object detectors use a resolution similar to, or lower than, YOLOv3.

6.3 Research Questions

This section attempts to answer each of the research questions in regard to the results and discussion.

RQ1: *How well does YOLOv3 locate sheep in UAV footage?*

The best configuration is the one that uses a superclass and a resolution of 832x832 pixels. At a threshold of 0.1, it is able to achieve a precision of 0.94 and a recall of 0.99 with only 12 out of 1650 sheep not being detected.

RQ2: *Is it easier to classify sheep as a superclass, or as one of three subclasses based on colour?*

Yes, it is easier to classify sheep as a superclass rather than as a subclass based on the performance achieved by the different configurations. All configurations with a superclass were equal or superior to the subclasses. It is worth to note that while a superclass performed best, peak performance for the subclasses was on average reached 1000 epochs before its counterpart.

RQ3: *Is it possible to increase performance noticeably by tweaking parameters post training?*

Parameter tweaking was limited to adjusting threshold and resolution. By modifying threshold it was possible to increase focus on either precision or recall, with recall being the highest at the lower threshold. Increasing the resolution was found to improve overall performance, however, the highest performance was not achieved by the highest resolution, but rather by the one in the middle: 832x832 pixels. This was consistent for both superclass and subclasses.

6.4 Threats to Validity

Threats to validity consist of both internal and external aspects of the experiments, data, and procedures that might have caused some interference or otherwise affected the results in an unintended or inevitable way.

6.4.1 Generalisability

The generalisability of the network refers to how well it may be applied to new and unknown data. In total there are three major issues to generalisability in this thesis:

Sheep distribution - The distribution of sheep is approximately 84%, 13% and 2% for white, black, and brown sheep respectively. Due to the low amount of black and brown sheep as compared to white sheep, the network is likely to perform significantly better on the localisation of white sheep. While it is estimated that white sheep constitute the largest portion of sheep in Norway, this would typically always result in datasets containing a

lot of white sheep. This is, unfortunately, counter-intuitive to the idea of training neural networks by example; the sheep expected to be hardest to find have the smallest data basis, which should result in poor performance.

Environment - The environment of the footage mainly consists of grassy fields. While there are some images from more typical Norwegian highlands, these do not contain any sheep and are simply negative samples in the dataset. It is unknown how well YOLOv3 will perform on any other kind of environment than these grassy fields, and it is not unlikely that the network has learnt to connect sheep to spots of white, brown and black surrounded by grass.

Contiguous Data - While data is technically from three different flights, the footage itself is quite similar, and the fact that images were taken at two-second intervals means that the same sheep are often present in multiple consecutive images. Imagine an image series of 5 images; image 1,2,4 and 5 are used to train a neural network, and image 3 is used to evaluate its performance. While the performance will likely be exceptional, it is unlikely that it will perform even remotely as well on data from another dataset. As neural networks learn from similar examples, be it from colour, shape, orientation, size or other, it is important to have a diverse dataset to hope to achieve any kind of generalisability.

6.4.2 Overfitting and Underfitting

When working with ANN's in general, overfitting and underfitting are two concerns that govern how long one should train the network. YOLOv3 has already proven to be a good model, however, if not trained long enough, or too long, it might perform too poorly to be useful. On the other hand, we have overfitting; this occurs when YOLOv3 has been trained for too long, and while it may perform exceptionally well on the training set, it is incapable of applying what has been learned to new data. It is next to impossible to define what the optimal amount of training is, and while it may be possible to improve results by training the network further, a decision was made to stick with 12000 epochs and leave further optimisations to other projects.

6.4.3 Erroneous Bounding Boxes

In total, four different issues were encountered that directly related to bounding box predictions:

Close Proximity - Multiple instances were discovered where sheep in close proximity to each other were often mislabelled as fewer sheep than were actually present, see Figure 6.3. This occurred to several different colours of sheep and is as such likely a larger problem than on related the shade of sheep. Both training and test sets contain multiple instances of sheep that are so close to each other that it is hard to determine where one sheep ends and the other begins. The issue might be caused by the way these sheep were labelled; all bounding boxes are rectangular and parallel to the corresponding image axis,

Figure 6.3: Example of bounding box issue cause by close proximity of two or more sheep. From *DJI_0313.JPG* with superclass at 1024x1024 pixels and a threshold of 0.1. Approximately the exact same issue was found in the corresponding subclass configuration as well.

which could result in some overlap when labelling sheep close to each other that were not oriented parallel to the image axes themselves.

Overlap - In a surprisingly large portion of images, multiple bounding boxes were predicted at highly overlapping locations, an instance of which is shown in Figure 6.4. YOLOv3 uses non-maximum suppression to automatically select the best bounding box when this kind of overlap occurs, however, it seems to be performing sub-par on this dataset. While YOLOv3 supports overlapping objects, in this case, sheep, this issue should only be widespread with the subclasses as multiple different coloured sheep may be predicted at the same location. This is however not the case, and the issue is similarly present in both superclass and subclasses. By increasing the threshold for detection most of these overlapping bounding boxes are discarded, albeit at the cost of a lower recall.

Complete Miss - The final issue is completely faulty predictions; a series of instances exist where YOLOv3 labels a patch of grass with no specific pattern, rocks, and outliers at the borders of images. The sizes appear entirely arbitrary and are not related to the size of correctly detected sheep in images. The issue is present in all tested configurations and likely consist of detections that barely pass the 0.1 objectness threshold limit. An example is shown in Figure 6.5.

6.4.4 Suggested Remedies to Increase Validity

Most, if not all, of the threats to validity, should be solved by using additional data for training. Data should represent as wide a range of environments and contexts as the desired use cases. Furthermore, increasing resolution during training to 832x832 or even 1024x1024 pixels could results in even more accurate predictions as showcased by having a network trained on images between 608x608 and 896x896 pixels.

Figure 6.4: Example of overlapping bounding boxes in output. From *DJI_0405.JPG* with superclass at 832x832 pixels and a threshold of 0.1.

Figure 6.5: Example of a complete miss on patches of grass. From *DJI_0407.JPG* with superclass at 1024x1024 pixels and a threshold of 0.1.

Chapter 7

Further Work

This chapter gives an overview of ideas considered for this project that were discarded due to being outside of the desired scope.

7.1 On-board Graphics

Nvidia has a series of portable modules that may be capable of performing analysis at a sufficient speed. The portable hardware does not need to be capable of training on its own, it should simply be able to utilise a pre-trained network at a reasonable speed.

7.2 Combine Infrared and Visual Imagery

While this thesis demonstrates the capability of YOLOv3 on visual imagery, the efficiency of YOLOv3 on Infrared still remains to be seen. The success of ordinary images advocates a success for infrared as well. There is also a possibility of using infrared images as a form of pruning; every image without a hotspot can be discarded as the infrared should be able to pick up everything remotely warmer than the background. As rocks in direct sunlight might also be picked up, by using YOLOv3 on the corresponding visual image it should be possible to verify the presence of sheep.

7.3 Use Altitude Data

As shown in Chapter 4, altitude measurements from the barometer appear reasonably accurate. It should be possible to use these measurements to prune out detected sheep that

are either too large or too small for what their expected size should be at a certain height. While it will not improve recall, it could definitely be used to achieve noticeably fewer false positives thus improving precision. As YOLOv3 is a fully convolutional network it is not possible to simply input height data next to the images; additional layers will likely have to be added as convolutional layers are not designed to handle data of this format.

7.4 Predators

It should be possible to extend the usage area of the developed solution to locate other kinds of animals as well, and predators are of particular interest to Norwegian farmers. A portion of sheep each year are lost to predators and the government compensates farmers for the loss in revenue if there is documentation that sheep are properly taken care of [9]. Additional tools to monitor the local predator populations could help with preparing appropriate preventive measures.

7.5 Improvements on YOLOv3

While working on this thesis, a new alternative to the YOLOv3 architecture was released; YOLOv3-SPP (Spatial Pyramid Pooling). This architecture increases the $mAP@50$ from 57.9 to 60.6 while maintaining the low inference time [24]. By simply using the same data and default parameters used in this thesis, it could be possible to increase performance or simply decrease the bounding box issues experienced.

Chapter 8

Conclusion

The primary objective for this thesis was to evaluate the performance of YOLOv3 on drone footage of sheep and secondary objectives consisted of tweaking parameters to better fit the dataset. The desired application was to use YOLOv3 to aid farmers in analysing drone footage from roundup at the end of the grazing period.

YOLOv3 performs exceptionally well on the drone footage and reaches peak performance of 99% recall and 94% precision, leaving out only 12 out of 1650 from detection. Detecting sheep as a superclass marginally outperformed detecting sheep as brown, black or white and it was possible to increase performance by increasing the network resolution. All configurations trained were capable of outperforming the reported performance of YOLOv3.

While the results are good, the applicability of the trained network is uncertain, however, likely negatively impacted by the limited variety of environments and conditions that are present in the footage. Additional data should be used for training before the trained network is applicable for practical uses.

The research could benefit from further work that examines ways to perform detection onboard a drone in real-time, and the use of infrared cameras as well. The literature and results also indicate that there are additional usages for YOLOv3 as an object detector, especially in regards to monitoring native predators.

Bibliography

[1] 2d convolution. https://github.com/PetarV-/TikZ/tree/master/2D Convolution. Accessed 27.02.2019.

[2] AlexeyAB. Darknet. https://github.com/AlexeyAB/darknet. Accessed 01.02.2019.

[3] AlexeyAB. Darknet. https://github.com/AlexeyAB/darknet#how-to-improve-object-detection. Accessed 01.02.2019.

[4] B. Benjdira, T. Khursheed, A. Koubaa, A. Ammar, and K. Ouni. Car Detection using Unmanned Aerial Vehicles: Comparison between Faster R-CNN and YOLOv3, 2018.

[5] U. Braga-Neto. Automatic target detection and tracking in forward-looking infrared image sequences using morphological connected operators. *Journal of Electronic Imaging*, 13(June 2003):1–22, 2004.

[6] L.-P. Chrétien, J. Théau, and P. Ménard. Wildlife Multispecies Remote Sensing Using Visible and Thermal Infrared Imagery Acquired From an Unmanned Aerial Vehicle (Uav). *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XL-1/W4:241–248, 2015.

[7] Common objects in context. http://cocodataset.org/#termsofuse. Accessed 20.05.2019.

[8] Dji mavic pro specification sheet. https://www.dji.com/no/mavic/specs. Accessed 15.10.2018.

[9] Fanesak tap av sau på beite. https://www.dyrebeskyttelsen.no/tap-sau-pa-beite/. Accessed 29.05.2019.

[10] Findmy. https://www.findmy.no/. Accessed 11.03.2019.

[11] J. Gallego, A. Pedraza, S. Lopez, G. Steiner, L. Gonzalez, A. Laurinavicius, and

G. Bueno. Glomerulus Classification with Convolutional Neural Networks - Scientific Figure on ResearchGate. https://www.researchgate.net/figure/AlexNet-CNN-architecture-layers_fig1_318168077. Accessed 18.03.2019.

[12] L. F. Gonzalez, G. A. Montes, E. Puig, S. Johnson, K. Mengersen, and K. J. Gaston. Unmanned aerial vehicles (UAVs) and artificial intelligence revolutionizing wildlife monitoring and conservation. *Sensors (Switzerland)*, 16(1), 2016.

[13] B. Hansen. Varmesøkende droner finner sauen. http://gardsdrift.no/varmes%C3%B8kende-droner-finner-sauen, 2015. Accessed 15.10.2018.

[14] J. Hui. Real-time Object Detection with YOLO, YOLOv2 and now YOLOv3. https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088. Accessed 01.02.2019.

[15] S.-O. Hvasshovd. Droner og Sau og litt til !! - Anvendelser og Muligheter. https://www.fylkesmannen.no/contentassets/cbf122460efa4e37a051c17c07fade0d/droner-buskerud-2017.pdf, 2017. Accessed 13.03.2019.

[16] A. Kathuria. What's new in yolo v3? https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b. Accessed 25.03.2019.

[17] S. V. Lab. Imagenet. http://image-net.org. Accessed 25.04.2019.

[18] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal Loss for Dense Object Detection, 8 2017.

[19] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. SSD: Single Shot MultiBox Detector, 12 2015.

[20] A. L.J. and M. I. The Norwegian Sheep Farming Production System. In M.-F. P. and R. R., editors, *Systems of sheep and goat production: Organization of husbandry and role of extension services*, volume 38 of *Options Méditerranéennes : Série A. Séminaires Méditerranéens*, pages 249–253. Zaragoza : CIHEAM, 1999.

[21] M. R. Minar and J. Naher. Recent Advances in Deep Learning: An Overview, 7 2018.

[22] I. R. Batch Normalization - Speed up Neural Network Training. https://medium.com/@ilango100/batch-normalization-speed-up-neural-network-training-245e39a62f85. Accessed 25.03.2019.

[23] M. Radovic, O. Adarkwa, and Q. Wang. Object Recognition in Aerial Images Using Convolutional Neural Networks. *Journal of Imaging*, 3(2):21, 2017.

[24] J. Redmon. Yolo: Real-time object detection. https://pjreddie.com/darknet/yolo/. Accessed 19.04.2019.

[25] J. Redmon. yolov3.cfg. https://github.com/pjreddie/darknet/blob/master/cfg/yolov3.cfg. Accessed 01.02.2019.

[26] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788. IEEE, 2016.

[27] J. Redmon and A. Farhadi. YOLO9000: Better, Faster, Stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525. IEEE, 2017.

[28] J. Redmon and A. Farhadi. YOLOv3: An Incremental Improvement. Technical report, University of Washington, 2018.

[29] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[30] Encyclopedia of life - sheep. http://eol.org/pages/311906/data. Accessed 15.10.2018.

[31] Shiip. https://www.shiip.no/. Accessed 11.03.2019.

[32] Smartbjella. https://smartbjella.no/. Accessed 11.03.2019.

[33] R. Smit. *Automatic animal detection using unmanned aerial vehicles in natural environments*. PhD thesis, University of Groningen, The Netherlands, 2016.

[34] J. Su, D. V. Vargas, and S. Kouichi. One pixel attack for fooling deep neural networks, 2017.

[35] P. Svenmarck, L. Luotsinen, M. Nilsson, and J. Schubert. Possibilities and challenges for artificial intelligence in military applications. In *Proceedings of the NATO Big Data and Artificial Intelligence for Military Decision Making Specialists' Meeting*, 2018.

[36] Telespor. https://telespor.no/. Accessed 11.03.2019.

[37] Tzutalin. Labelimg. https://github.com/tzutalin/labelImg. Accessed 20.10.2018.

[38] J. C. van Gemert, C. R. Verschoor, P. Mettes, K. Epema, L. P. Koh, and S. Wich. Nature conservation drones for automatic localization and counting of animals. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 8925, pages 255–270, 2015.

[39] O. Vangen and A. Blix. Sau. https://snl.no/sau, 2018. Accessed 12.03.2019.

[40] Y. Zhong, J. Wang, J. Peng, and L. Zhang. Anchor Box Optimization for Object Detection, 2018.

Appendix A - DJI Mavic Pro Field of View

Gmail **Jonas Muribø <jmurib@gmail.com>**

[Support] : RE Field of view for Mavic Pro

James Russell (DJI Support) <support@dji.com> 17 October 2018 at 12:09
Reply-To: DJI Support <support@dji.com>
To: Jonas Muribø <jmurib@gmail.com>

##- 请在此行以上输入您的回复 -##

Your request（#1416766）has been updated, please reply the email below.

James Russell （Support）
10月17日 CST18:09

Hello Jonas,

Good day!

This is to inform you that our engineers provide us an answer about your query.
The FOV is 66.8° in horizontally, 52.5° in vertically.

If you have any further questions, feel free to contact us.

[Quoted text hidden]

[Quoted text hidden]

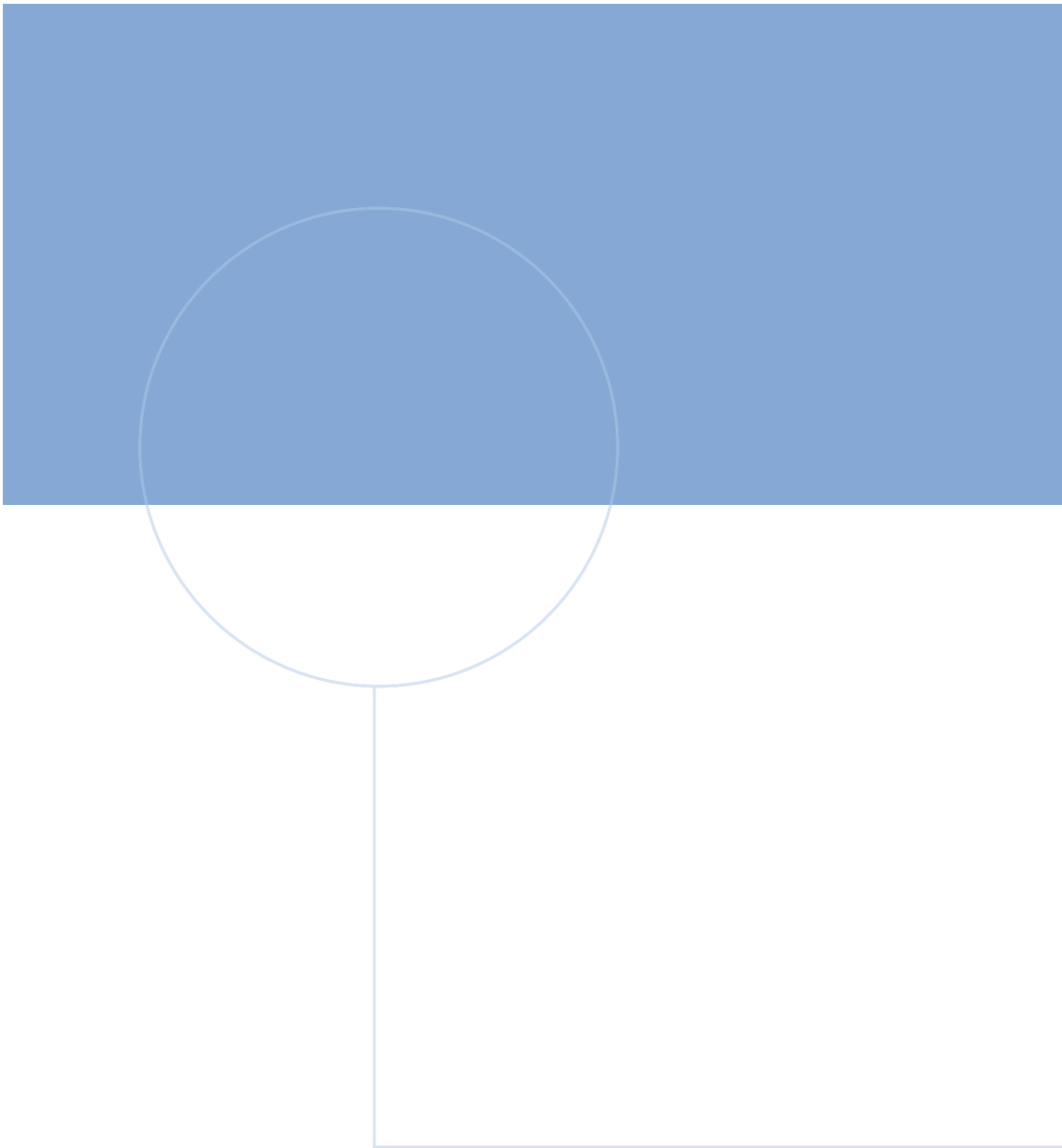
[Quoted text hidden]

[YD6PGL-R4D7]

Appendix B - YOLOv3 Layers

```
layer     filters    size              input                output
    0 conv     32  3 x 3 / 1   416 x 416 x   3   ->   416 x 416 x  32
    1 conv     64  3 x 3 / 2   416 x 416 x  32   ->   208 x 208 x  64
    2 conv     32  1 x 1 / 1   208 x 208 x  64   ->   208 x 208 x  32
    3 conv     64  3 x 3 / 1   208 x 208 x  32   ->   208 x 208 x  64
    4 Shortcut Layer: 1
    5 conv    128  3 x 3 / 2   208 x 208 x  64   ->   104 x 104 x 128
    6 conv     64  1 x 1 / 1   104 x 104 x 128   ->   104 x 104 x  64
    7 conv    128  3 x 3 / 1   104 x 104 x  64   ->   104 x 104 x 128
    8 Shortcut Layer: 5
    9 conv     64  1 x 1 / 1   104 x 104 x 128   ->   104 x 104 x  64
   10 conv    128  3 x 3 / 1   104 x 104 x  64   ->   104 x 104 x 128
   11 Shortcut Layer: 8
   12 conv    256  3 x 3 / 2   104 x 104 x 128   ->    52 x  52 x 256
   13 conv    128  1 x 1 / 1    52 x  52 x 256   ->    52 x  52 x 128
   14 conv    256  3 x 3 / 1    52 x  52 x 128   ->    52 x  52 x 256
   15 Shortcut Layer: 12
   16 conv    128  1 x 1 / 1    52 x  52 x 256   ->    52 x  52 x 128
   17 conv    256  3 x 3 / 1    52 x  52 x 128   ->    52 x  52 x 256
   18 Shortcut Layer: 15
   19 conv    128  1 x 1 / 1    52 x  52 x 256   ->    52 x  52 x 128
   20 conv    256  3 x 3 / 1    52 x  52 x 128   ->    52 x  52 x 256
   21 Shortcut Layer: 18
   22 conv    128  1 x 1 / 1    52 x  52 x 256   ->    52 x  52 x 128
   23 conv    256  3 x 3 / 1    52 x  52 x 128   ->    52 x  52 x 256
   24 Shortcut Layer: 21
   25 conv    128  1 x 1 / 1    52 x  52 x 256   ->    52 x  52 x 128
   26 conv    256  3 x 3 / 1    52 x  52 x 128   ->    52 x  52 x 256
   27 Shortcut Layer: 24
   28 conv    128  1 x 1 / 1    52 x  52 x 256   ->    52 x  52 x 128
   29 conv    256  3 x 3 / 1    52 x  52 x 128   ->    52 x  52 x 256
   30 Shortcut Layer: 27
   31 conv    128  1 x 1 / 1    52 x  52 x 256   ->    52 x  52 x 128
   32 conv    256  3 x 3 / 1    52 x  52 x 128   ->    52 x  52 x 256
   33 Shortcut Layer: 30
   34 conv    128  1 x 1 / 1    52 x  52 x 256   ->    52 x  52 x 128
   35 conv    256  3 x 3 / 1    52 x  52 x 128   ->    52 x  52 x 256
   36 Shortcut Layer: 33
   37 conv    512  3 x 3 / 2    52 x  52 x 256   ->    26 x  26 x 512
   38 conv    256  1 x 1 / 1    26 x  26 x 512   ->    26 x  26 x 256
   39 conv    512  3 x 3 / 1    26 x  26 x 256   ->    26 x  26 x 512
   40 Shortcut Layer: 37
   41 conv    256  1 x 1 / 1    26 x  26 x 512   ->    26 x  26 x 256
   42 conv    512  3 x 3 / 1    26 x  26 x 256   ->    26 x  26 x 512
   43 Shortcut Layer: 40
   44 conv    256  1 x 1 / 1    26 x  26 x 512   ->    26 x  26 x 256
   45 conv    512  3 x 3 / 1    26 x  26 x 256   ->    26 x  26 x 512
   46 Shortcut Layer: 43
   47 conv    256  1 x 1 / 1    26 x  26 x 512   ->    26 x  26 x 256
   48 conv    512  3 x 3 / 1    26 x  26 x 256   ->    26 x  26 x 512
   49 Shortcut Layer: 46
   50 conv    256  1 x 1 / 1    26 x  26 x 512   ->    26 x  26 x 256
   51 conv    512  3 x 3 / 1    26 x  26 x 256   ->    26 x  26 x 512
   52 Shortcut Layer: 49
```

```
53 conv    256  1 x 1 / 1    26 x  26 x 512   ->    26 x  26 x 256
54 conv    512  3 x 3 / 1    26 x  26 x 256   ->    26 x  26 x 512
55 Shortcut Layer: 52
56 conv    256  1 x 1 / 1    26 x  26 x 512   ->    26 x  26 x 256
57 conv    512  3 x 3 / 1    26 x  26 x 256   ->    26 x  26 x 512
58 Shortcut Layer: 55
59 conv    256  1 x 1 / 1    26 x  26 x 512   ->    26 x  26 x 256
60 conv    512  3 x 3 / 1    26 x  26 x 256   ->    26 x  26 x 512
61 Shortcut Layer: 58
62 conv   1024  3 x 3 / 2    26 x  26 x 512   ->    13 x  13 x1024
63 conv    512  1 x 1 / 1    13 x  13 x1024   ->    13 x  13 x 512
64 conv   1024  3 x 3 / 1    13 x  13 x 512   ->    13 x  13 x1024
65 Shortcut Layer: 62
66 conv    512  1 x 1 / 1    13 x  13 x1024   ->    13 x  13 x 512
67 conv   1024  3 x 3 / 1    13 x  13 x 512   ->    13 x  13 x1024
68 Shortcut Layer: 65
69 conv    512  1 x 1 / 1    13 x  13 x1024   ->    13 x  13 x 512
70 conv   1024  3 x 3 / 1    13 x  13 x 512   ->    13 x  13 x1024
71 Shortcut Layer: 68
72 conv    512  1 x 1 / 1    13 x  13 x1024   ->    13 x  13 x 512
73 conv   1024  3 x 3 / 1    13 x  13 x 512   ->    13 x  13 x1024
74 Shortcut Layer: 71
75 conv    512  1 x 1 / 1    13 x  13 x1024   ->    13 x  13 x 512
76 conv   1024  3 x 3 / 1    13 x  13 x 512   ->    13 x  13 x1024
77 conv    512  1 x 1 / 1    13 x  13 x1024   ->    13 x  13 x 512
78 conv   1024  3 x 3 / 1    13 x  13 x 512   ->    13 x  13 x1024
79 conv    512  1 x 1 / 1    13 x  13 x1024   ->    13 x  13 x 512
80 conv   1024  3 x 3 / 1    13 x  13 x 512   ->    13 x  13 x1024
81 conv     18  1 x 1 / 1    13 x  13 x1024   ->    13 x  13 x  18
82 detection
83 route  79
84 conv    256  1 x 1 / 1    13 x  13 x 512   ->    13 x  13 x 256
85 upsample           2x    13 x  13 x 256   ->    26 x  26 x 256
86 route  85 61
87 conv    256  1 x 1 / 1    26 x  26 x 768   ->    26 x  26 x 256
88 conv    512  3 x 3 / 1    26 x  26 x 256   ->    26 x  26 x 512
89 conv    256  1 x 1 / 1    26 x  26 x 512   ->    26 x  26 x 256
90 conv    512  3 x 3 / 1    26 x  26 x 256   ->    26 x  26 x 512
91 conv    256  1 x 1 / 1    26 x  26 x 512   ->    26 x  26 x 256
92 conv    512  3 x 3 / 1    26 x  26 x 256   ->    26 x  26 x 512
93 conv     18  1 x 1 / 1    26 x  26 x 512   ->    26 x  26 x  18
94 detection
95 route  91
96 conv    128  1 x 1 / 1    26 x  26 x 256   ->    26 x  26 x 128
97 upsample           2x    26 x  26 x 128   ->    52 x  52 x 128
98 route  97 36
99 conv    128  1 x 1 / 1    52 x  52 x 384   ->    52 x  52 x 128
100 conv   256  3 x 3 / 1    52 x  52 x 128   ->    52 x  52 x 256
101 conv   128  1 x 1 / 1    52 x  52 x 256   ->    52 x  52 x 128
102 conv   256  3 x 3 / 1    52 x  52 x 128   ->    52 x  52 x 256
103 conv   128  1 x 1 / 1    52 x  52 x 256   ->    52 x  52 x 128
104 conv   256  3 x 3 / 1    52 x  52 x 128   ->    52 x  52 x 256
105 conv    18  1 x 1 / 1    52 x  52 x 256   ->    52 x  52 x  18
106 detection
```



NTNU
Norwegian University of
Science and Technology