Master's thesis

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

Hallvard Stemshaug

Impact of Low Resolution IR Images in Drone Based Sheep Detection

Master's thesis in Computer Science
Supervisor: Svein-Olaf Hvasshovd
June 2022

NTNU
Norwegian University of
Science and Technology

Hallvard Stemshaug

Impact of Low Resolution IR Images in Drone Based Sheep Detection

Master's thesis in Computer Science
Supervisor: Svein-Olaf Hvasshovd
June 2022

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

NTNU

Kunnskap for en bedre verden

Abstract

With a highly developed visual system, humans can recognize objects with high precision. The detection is based on visual cues such as shape, size, color, and motion. In recent years significant progress has been made in computer vision and deep learning. However, compared to the human visual system, the computer's toolbox is more limited, relying only on recognizing the visual features of the objects. With the availability of infrared cameras, features from the infrared spectrum can be added to the set of tools.

In this thesis, the topic is detecting sheep in images captured with the drone DJI Mavic 2 Enterprise Dual using Deep Learning-based object recognition. It is motivated by the aim of helping farmers whose method for finding sheep has traditionally been manual search. As a rule, such methods require a lot of preparation and effort, diminishing returns when fewer sheep are found.

This thesis explains fundamental theoretical concepts in computer vision, neural networks, deep learning, and dataset composition. Then, the methods, approach, and results of an experiment in the localization of sheep are presented using the object location model YOLOv5.

The key question of this research is to determine the impact of low-resolution single-channel IR (infrared) images used in addition to three-channel RGB (red, green, blue) color images on training a YOLOv5 model. A total of 18 models were trained and tested with different configurations based on splitting the images, using IR images in the data, and classifying sheep based on color.

The results showed that the model that achieved the highest detection rate at 98.6 % of sheep in the test set used full images, the original RGB images, and four different classes of sheep. The IR images had a slight negative impact on the detection of sheep. In comparison, the best performing model fully utilizing RGB and IR images was able to locate 96.0 % of the sheep in the test set.

The thesis concludes that it is not profitable for a farmer to invest in a low-resolution IR camera for sheep detection when camera drones with similar RGB image sensors exist at a much lower cost.

Sammendrag

Ved hjelp av en velutviklet synssans, klarer mennesker å se kjenne igjen og se forskjell på objekter med høy presisjon. Vi gjør dette basert på objektets form, størrelse eller farge og med ved å oppdage bevegelse. Det er gjort betydelige fremskritt innen datasyn og dyp læring de siste årene. Likevel er baserer datamaskinens synssans seg i de flest tilfeller på å gjenkjenne objektenes synlige kjennetegn. Med infrarøde kameraer lettere tilgjengelig kan dette synssystemet utvides til å inkludere data fra det infrarøde spekteret.

I denne oppgaven er temaet gjenfinning av sau i bilder tatt med dronen DJI Mavic 2 Enterprise Dual. Dette blir gjort ved bruk dyplærings basert objektgjenkjenning. Motivasjonen bak arbeidet er å hjelpe bønder med å effektivisere sauesankingen som tradisjonelt har vært basert på manuelt arbeid. Som regel krever sauesankinga mye forberedelse og innsats, med avtagende avkastning når det er færre sauer igjen å finne.

Denne oppgaven forklarer sentrale grunnleggende teoretiske konsepter innen datasyn, nevrale nettverk, dyp læring og datasettsammensetning. Deretter presenteres metodene, tilnærmingen og resultatene av et eksperiment i lokalisering av sau ved bruk av dyplæringsmodellen YOLOv5.

Målet i oppgaven er å fastslå hvilken effekt bruken av IR (infrarøde) bilder med lav oppløsning brukt sammen med fargebilder har på opptreningen av en YOLOv5-modell. I sum ble 18 modeller opptrent og testet med forskjellige konfigurasjoner av oppdeling av bildet, bruk av IR bildene i datasettet, og hvordan sauene ble delt in i kategorier etter farge.

Resultatene fra oppgaven viser at modellen som oppnådde den høyeste gjenkjennelseraten på 98.6 % for sauer i testsettet bruker hele bilder, RGB bilder alene, og fire forskjellige kategorier for sauene. Til sammenligning oppnådde modellen med de beste resultatene for kombinert bruk av RGB bilder og IR bilder en gjenkjennelserate på 96.0% sauer i testsettet.

Oppgaven konkluderte med at det ikke er lønnsomt for bønder å investere i et lavoppløselig IR-kamera for sauegjenfinning når kameradroner med lignende RGB-bildesensorer finnes til en mye billigere penge.

Acknowledgements

I would like to thank my supervisor, professor Svein-Olaf Hvasshovd for his guidance and encouragement.

Thank you to everyone who helped to expand the dataset of sheep images; may it be useful for future theses. Thanks to Kari Meling Johansen for sharing her dataset from previous experiments and to Bjørnar Østtveit, Sebastian Vittersø, and Ingebrigt Nygård for their collaboration with labeling the images.

Thanks to Svartådalen Beitelag for bringing me along on their sheep gathering, and a special thank you to Line Buan, who guided me through the forest so that I could find and photograph her sheep.

The biggest thanks and appreciation to the HPC group at IDI for giving me access to the IDUN cluster [1]. Without access to this great resource, I could not have conducted my research.

Finally, I would like to thank my dad for sharing his forty years of experience as a sheep farmer. Our work would be done if a computer could ever match his ability to tell sheep apart from each other.

Table of Contents

List of Figures

List of Tables

Nomenclature

AI	Artificial Intelligence
AP	Average Precision
$CIoU$	Complete IoU
CLS	Class Label Smoothing Regularization
CNN	Convolutional Neural Network
$COCO$	Common Objects in Context
DL	Deep Learning
FN	False Negative
FP	False Positive
$GIoU$	Generalized IoU
GPU	Graphics Processing Unit
IoU	Intersection over Union
mAP	Mean Average Precision
ML	Machine Learning
NMS	Non-Maximum-Suppression
$PANet$	Path Aggregation Network
$R-CNN$	Region-based Convolutional Neural Networks
$ReLU$	Rectified Linear Unit
SGD	Stochastic Gradient Descent
SPP	Spatial Pyramid Pooling
TP	True Positive
$YOLO$	You Only Look Once

Chapter 1

Introduction

Ever since sheep have been kept as livestock, monitoring the animals has been a challenge to overcome. Shepherds had to depend on their senses and keen eyesight to keep track of the herd and look out for dangerous predators. The human eye is an incredible visual perception system. In the first years of life, humans learn to distinguish objects by their shape and color while identifying people by their facial features, skin, and hair color. Within fractions of a second, we remember new objects and places and can recognize them days, weeks, or even years after we first saw them. The human visual system also enables us to search, locate and identify various objects in our environment with high precision [2].

With significant advances in computer vision and deep learning (DL) in recent years, recognizing objects in their natural environment is still challenging for computers. Object detection aims to teach computers something close to or superior to "human vision," where objects are to be recognized in optical inputs and classified as human-understandable object classes. The type of object to be detected depends on the specific task and is, in most applications, not general-purpose like the human visual system [3, 4].

The increased commercial availability of camera drones has led to promising results in using footage captured from unmanned aerial vehicles (UAV) in combination with computer vision [5]. They are used in various applications, including search and rescue operations and facility inspection. It has also shown promise in the zoological and agricultural fields, with applications in wildlife monitoring [6, 7], livestock counting[8], and weed detection in crops [9].

1.1 Background and Problem definition

Each spring in Norway, the farmers release about two million sheep to graze in mountain and forest areas, where they are left primarily unsupervised. The sheep must be rounded up and brought back to the farm in the fall at the end of the grazing season [10]. With grazing areas often spanning multiple square kilometers, this process is time-consuming and requires a large workforce of farmers and volunteers. In addition, it is not common that they find all sheep in the first search, which usually leads to additional rounds throughout the fall. At the end of the season, farmers rely on observations from hikers and hunters to find the still missing sheep. If the farmers cannot find them by the end of the fall, they will be considered lost.

When searching for sheep, farmers often use binoculars to survey the area from a vantage point and use walkie-talkies to explain where they observed sheep. To narrow the area to be searched, they often precisely select areas where they have observed sheep during the summer or know from experience that sheep are usually present. This strategy requires knowing both the area and the sheep's behavior patterns.

Recent developments in research and technology enable data collection with unprecedented accur-

acy and dimension today. In particular, UAVs, specifically drones, have significantly advanced data collection. Drones make it possible to quickly and reliably collect aerial photography from pastures and thus form a reliable basis for locating animals [6]. However, the central part of the work is not in the data collection itself but in identifying the sheep in the images. Manual identification would be extremely time-consuming due to the large amounts of data. One way to simplify sheep detection is to use an image recognition model trained using machine learning. Image recognition models for automatic sheep detection offer the possibility to process large data sets and thus reduce the manual workload for the farmer.

Creating a robust image recognition model comes with its own set of challenges. For example, one requirement for locating sheep in the mountains is that the drone captures images of where the sheep roam. The images must then be analyzed using an object recognition model fine-tuned for sheep. In this process, the sheep grazing environment will not be constant; each location has different landscapes whose surroundings change depending on the season and weather conditions. For this reason, an image recognition model faces the challenges of the changing environment. In order to accurately locate and recognize sheep, the image recognition model must be able to deal with a wide variety of landscapes, light and weather conditions, as well as differences in the appearance of different breeds of sheep and individual differences within those breeds.

1.2 Objectives

This thesis focuses on applying machine learning to detecting sheep using a combination of color and thermal aerial images captured by drones. There is a significant price difference between commercially available drones equipped with both color and thermal cameras and drones with a single comparable color camera. For this reason, the goal is to gain insight into whether thermal imagery is advantageous compared to color-only imagery.

Building on previous research on sheep detection in drone imagery at NTNU, this work aims to investigate which factors affect the performance of the model, which methods and strategies need further development, and which ones have low priority in the current state of the art. Although other master's theses have created machine learning models that locate sheep with high accuracy, it remains unclear how much the IR imagery affected the model's performance. This thesis aims to provide a clearer picture of how the composition of the dataset impacts the model's success, how significant the impact of IR images is on the final result, and whether the use of currently available IR technology is worth the investment.

This process involves creating a dataset of aerial images of sheep to train artificial neural network (ANN) models that apply to similar images captured in the area where the sheep are out to pasture during the gathering season. The ANN's task is to locate the sheep accurately to be reliably recognized and classified. An accurate and operational machine learning model for sheep detection creates the possibility of interaction between humans and machines, where they support each other instead of acting independently. Furthermore, once an efficient model is found, the farmer's workload can be significantly reduced and its complexity simplified by using a system that helps the farmer save time and effort in collecting the sheep from the pasture.

1.3 Structure of the thesis

This thesis has seven chapters. Chapter 1 introduces the background and challenges farmers face in collecting their sheep today. Furthermore, it deals with the objective of this thesis and its structure.

Chapter 2 provides an overview of the state of the art and related work that forms the foundation of this thesis.

Chapter 3 introduces the basics of computer vision and deep learning. This section introduces the concepts of ANNs and deep neural network architecture. Next, the chapter describes various

methods for model evaluation, including the YOLOv5 method, which forms the basis for the data evaluation in this work. Finally, there is a discussion of possible problems in training such a network and a review of the data set's composition.

Chapter 4 covers the requirements and steps to develop, train, and evaluate the models for sheep detection. This chapter applies the relevant methods and discusses data acquisition. In addition, this work investigates the image and label preprocessing steps and the applied machine learning model.

Chapter 5 shows the result of training an image recognition model on three different variables. The first is if the images are full size or split into patches with background images removed or pruned. The second variable is how the IR image is used; not used, merged with the color image, and concatenated to the color image. The final variable is how many classes the sheep are divided into, four, two, and one. Finally, there is an evaluation of the best-performing models by the larger YOLOV5 network and comparing the different methods.

Chapter 6 reviews and critically discusses the validity and implications of the findings presented in the previous chapter and an evaluation of the results concerning the research questions.

Finally, in Chapter 7, the conclusion and future work summarize the work and elaborates on their validity. Then, based on the results of this work, recommendations for future work are outlined.

Chapter 2

State of the Art

This chapter discusses previous work related to this thesis at NTNU and other research efforts on using drones and IR imagery for animal detection. Finally, an overview of existing and competing sheep tracking and gathering technologies will be presented.

2.1 Earlier Master Theses

In recent years master's students at NTNU supervised by Prof. Svein-Olaf Hvasshovd have written numerous dissertations on sheep recognition. There have been attempts to use both DL [11] [12] [13] [14] as well as traditional computer vision technologies. Since this research deals with DL, special attention is given to the former.

Muribøs thesis from 2019 [11] used a YOLOv3 type network to locate sheep. Using only color images, Muribø was able to locate over 99% of sheep in the test set. While showing the promise of YOLO-type networks in detecting sheep, the author points out some flaws in his methodology. The sheep were considered found if they had a confidence score over 0.1, without any attempts at correcting for multiple detections of the same sheep, boosting the rate of detections. It was also pointed out that the variation of the dataset was lacking. The images were taken under similar lighting conditions and in a similar environment with sheep in the pasture. Approximately 84% of the sheep were white. Muribø recommends using a much more diverse data set for future work.

In 2019 Guttormsen [15] wrote an extensive master thesis about data collection using a drone with a thermal camera. He reported that the difference between the sheep's surface temperature and the ground temperature was, on average, 3.89 ° C. The highest temperature difference was reported as 7.75 ° C and the lowest as 2 ° C.

Building on this, Johansen, in her thesis from 2020 [12] trained a fusion network with a network for RGB images and a network for IR images, whose results she merged in a fusion step. In this way, both RGB and IR images could be used. In addition, the localization problem was simplified to a classification problem where sheep are localized in a grid of 4x4 cells. With a confidence threshold of 0.5, the most accurate network achieved a precision of 97.7%. In the grid cells, it achieved a recall of 90.1% and detected 97.5% of the sheep in the validation dataset. Johansen notes that while the network performed well in the test dataset, primarily brown sheep, underrepresented, performed worse. Furseth and Granås [13] followed up Johansen's research in 2021 by developing a system for running sheep detection models on mobile devices. They used the YOLOv5 network and split the training images into tiles to reduce the inference time on the mobile device. Their best model detected 98% of the sheep in their test set at an inference time of 8. In addition, they discovered that models trained with tiled images performed better when used on full images than those originally trained with full-sized images. The authors note that future work should attempt to expand the dataset and improve the image quality of the current dataset. In addition, they recommend that the acquisition strategy of IR image data be further investigated due to its

potential. This observation was made since the quality of IR images in the dataset they used was insufficient for their needs. In his research in 2021, Bøchman [14] reviewed the existing datasets used in the work as mentioned earlier using a lightweight detection system based on YOLOv3 Tiny. He found that the data collection approaches of the various students differed. He pointed out that this was likely to impact the performance of the networks and strongly suggested that further work be done to improve the dataset's quality.

2.2 Related research

More research using UAVs has emerged since affordable camera drones became commercially available. Outside of NTNU, limited research has been conducted on concepts for collecting and rounding up sheep in the mountains. However, approaches to counting sheep using UAVs have been investigated. There have also been various approaches to using UAVs for wildlife monitoring.

In their 2021 publication, Sarwar et al. propose using drones to detect and count sheep [8]. They trained a neural network based on images of sheep in a pasture from heights of 80 and 120 meters. The researchers proposed a network architecture, U-Net-MS, that received an F1 score of 98% on their test set. According to them, training the network with images taken from a different height than the test set significantly impacted its accuracy. In contrast, a training set with images from both altitudes resulted in the network performing well on both tasks. In addition, the researchers used only color images to train their model. While this study showed that the sheep could be identified accurately, the dataset consisted of sheep images in an open pasture. Thus, it was not taken in challenging terrain or areas where the sheep were covered by vegetation.

Gonzalez et al. proposed a non-invasive method to conduct the survey compared to traditional methods in their 2016 paper [6]. Their system used machine learning to train a model on IR imagery captured by UAVs to detect threatened and invasive species and produce a population estimate. As a result, the researchers successfully detected targeted wildlife in their studied area.

Rey et al. [7] propose a system for monitoring wildlife and livestock in the semi-arid African savanna using images captured by cameras mounted on fixed-wing drones. Their suggestion is based on a semi-automatic system in which the images are processed by machine learning. Afterward, the results were analyzed by a human operator to eliminate false positives. They proposed using the semi-automatic approach because the model with a high recognition rate was chosen, tolerating a higher number of false positives while minimizing the number of false negatives. The reasoning was that using a semi-automatic approach might be beneficial when the consequence of not finding something (false positive) is much higher than finding something that is not there (false negative).

In a 2021 paper on wildlife detection, Lee, Song, and Kil [16] uses a model that processes IR images with traditional computer vision methods based primarily on the Sobel edge detection algorithm. This model requires no training data. Their solution proposes using IR images as a lightweight supplement to the more expensive machine learning approaches. The researchers achieved a detection time of 0.033 seconds for their fastest approach. Their most accurate model achieved precision and recall of 0.804 and 0.699. Notably, they discovered that while it was possible to achieve usable results up to a flight height up to 100m, the quality of images captured with a low-resolution IR-image sensor deteriorates as the flight height increases.

2.3 Other technologies in use

While this thesis concerns detecting sheep with UAVs, there exists a set of other methods that farmers already use to keep track of their sheep. This section will give an overview of the different available tools.

2.3.1 Bells and ear tags

The use of bells is a simple but proven technique to monitor sheep. The bell is attached around the sheep's neck using a collar. As soon as the sheep moves, the bell makes a sound. This way, the person looking for the sheep can track it even if he loses sight of it. The sound of the bell can be heard over long distances and has been essential for locating sheep in larger areas. One disadvantage of the bell is that it is difficult to hear in noisy environments. In addition, the bell relies on the movement of sheep to produce noise. The problem is when the sheep are stationary, especially when they are sick or dead [17]. It can be debated whether carrying the bell causes discomfort to the sheep or makes them easier prey by alerting predators in close range to their location [18].

In addition to the bell, most sheep are marked with ear tags indicating their owner. It is not uncommon for sheep to intermix in areas close to or overlap with other sheep owners' pastures, despite being fenced. Ear tags prevent sheep from being rounded up when they should not be and end up with the wrong owner at the end of the season [19]. However, they often cause an inflammatory reaction and require proper positioning to minimize the severity of ear injuries [20].

2.3.2 Radio collars

There are several alternative ways to find sheep. GPS tracking collars, for example, have become increasingly popular in recent years. The following section reviews four vendors in this market: Nofence, Telespor, Findmy, and Smartbjella.

Nofence

Nofence is a system that trains animals to stay within an area set by the user. It has a collar with a GPS tracker and electroshock stimuli for the animals exiting the preset area. This device has been used with farm animals, especially goats, sheep, and cows. These animals quickly learn to respond to the boundary system and largely adhere to the predetermined area. Nofence also has tracking capabilities with a motion sensor and a GPS receiver to transmit the animals' position to the farmer [21]

The device is powered by solar energy to extend the battery's life. It is a lithium-ion battery, and one battery is reportedly enough for a six-month grazing season, depending on reception in the region. The total weight of the collar and battery is 505 grams [21].

Telespor

Radiobjella is a waterproof GPS collar from Telepor that tracks sheep using GNSS and LTE-M technology. The sheep's location is constantly updated, and the collar also records the history of the sheep's location. In addition, the collar contains a motion sensor activated when the sheep is at the exact position for an extended period and when the last two communications with the server have failed. When one of these alarms is triggered, the user is notified by SMS or email. The collar comes with a replaceable lithium battery, which Telespor recommends changing after each season. The combined weight of the collar and battery is 104 grams [22].

Findmy

E-Bjella, built by Findmy, is another GPS collar tracking solution for farm animals that allows the farmer to monitor the animals. A geofence function alerts when animals leave a user-defined area. According to the manufacturer, the collars communicate with low-orbit satellites with the best GPS coverage. They claim the collar's battery life lasts for 2-3 seasons with a grazing season

of six months. In addition, the system can track unrest in the sheep's movements and alert the farmer. For example, if there are predators near the flock [23].

Smartbjella

Smartbjella, similar to the previously mentioned solutions, is a collar for locating livestock and transmitting their location. As seen in Radiobjella, it also has a "death detection" feature triggered when an animal does not move for an extended time. However, it differs from the other solutions because it primarily tracks the animal using narrowband IoT and only uses GPS tracking when an accurate location is needed. As a result, Smartbjella promises a longer battery life than its competitors, depending on how often the collar is configured to update its location. The battery life promised is 1.5 years, with an update sent every hour. If updated daily, the life expectancy is 17 years. These numbers are based on the assumption that the animals are in an area with good cellular reception. Overall, the collar weighs 140 grams [24].

2.3.3 Comparison of UAVs from DJI

In 1979, even before drones were considered for sheep capture, Przybilla and Wester-Ebbinghaus used a radio-controlled model helicopter carrying a camera for aerial photography [25].

Today, there are numerous manufacturers of UAVs. One of the most successful in the field of UAVs for video and photography is a Chinese technology company founded by Frank Wang in 2006, Da-Jiang Innovations Science and Technology Co, Ltd, or DJI for short [26].

Meanwhile, a wide variety of drone models are offered by DJI. For this and past master theses at NTNU, the DJI Mavic Enterprise Dual was used. It is a thermal imaging drone with a dual thermal imaging camera called thermal, FLIR, or IR camera[27].

All models are so-called multicopters. During takeoff, the rotation speed of the individual propellers is increased simultaneously to the same extent. By varying the ambient speed of individual propellers, the drone can be easily maneuvered and rotated around its axis. These UAVs are controlled by software. They have a remote control with a screen or even a mount to connect the smartphone [26]. The following table 2.1 lists drones from DJI in different price ranges, camera resolution, flight time, and weight.

	Price	Camera resolution	Flight time	Weight
DJI Mavic 2 Enterprise Advanced	64.990 NOK	48MP	31 min	1100 g
DJI Mavic 2 Enterprise (Dual)	32.690 NOK	12MP	31 min	899 g
DJI Mavic 3	19.556 NOK	20MP	46 min	895 g
DJI Mavic Air 2	8.889 NOK	48 MP	34 min	570 g

Table 2.1: Comparison of UAVs

Chapter 3

Theory

3.1 Computer Vision and Deep Learning

Computer vision is an interdisciplinary field of research that focuses on developing models and methods for the machine acquisition, processing, and analysis of images or other high-dimensional data [3]. The approach increasingly relies on machine learning (ML), particularly DL. This process uses ANNs with complex, deeply nested network architectures to better recognize and process internal data representations at multiple levels of abstraction [28].

3.1.1 Types of detection

Object recognition identifies objects in an image or video taken in a typical human environment with a camera sensor operating in the optical spectrum. Although this is the most common use case, object recognition with DL is also used for images acquired with other image sensor techniques, such as thermal images [29] and close-range data acquired with LIDARS. In this context, an object class or category is a group of objects with similar visual properties. Objects that belong to a category are called instances of that category. In a narrow sense, it can be an object that is easily identifiable as an object, such as a car or a horse. In a broader sense, the instance can be anything with a visual representation in its own right, such as a risk area in the snow that can trigger avalanches[3].

Object recognition is an application area of Artificial Intelligence (AI) that has gained relevance in recent years as graphics processes have become more powerful. In addition, the emergence of cell phones has made more training data readily available. For example, autonomous driving enables the recognition of cars, pedestrians, or other obstacles. Other applications automatically identify tumors or anomalies in medicine and face recognition in biometrics, which is used in many smartphones today[3].

The basis of traditional object recognition algorithms has three main steps: segmentation, localization, and classification [3]. The first two steps are automatically performed in ANNs by the feature extractors and the last step by the network head. The steps are explained here because the concepts are also neural network terminology, and the presentation gives an insight into the tasks performed by a network in object detection[3].

Segmentation

Segmentation is the separation of objects from their background. It specifies the task of finding a group of pixels that belong together [**g**, 3]. Commonly applied algorithms use region splitting and merging, graph-based segmentation, and probabilistic aggregation [3, 30].

Feature extraction

In object localization, feature extraction plays a vital role in finding the features that define the objects. Examples include edges found by edge detection, corners found by corner detection, or other defining features that indicate the bounds or shape of a localized object[3, 31].

Classification

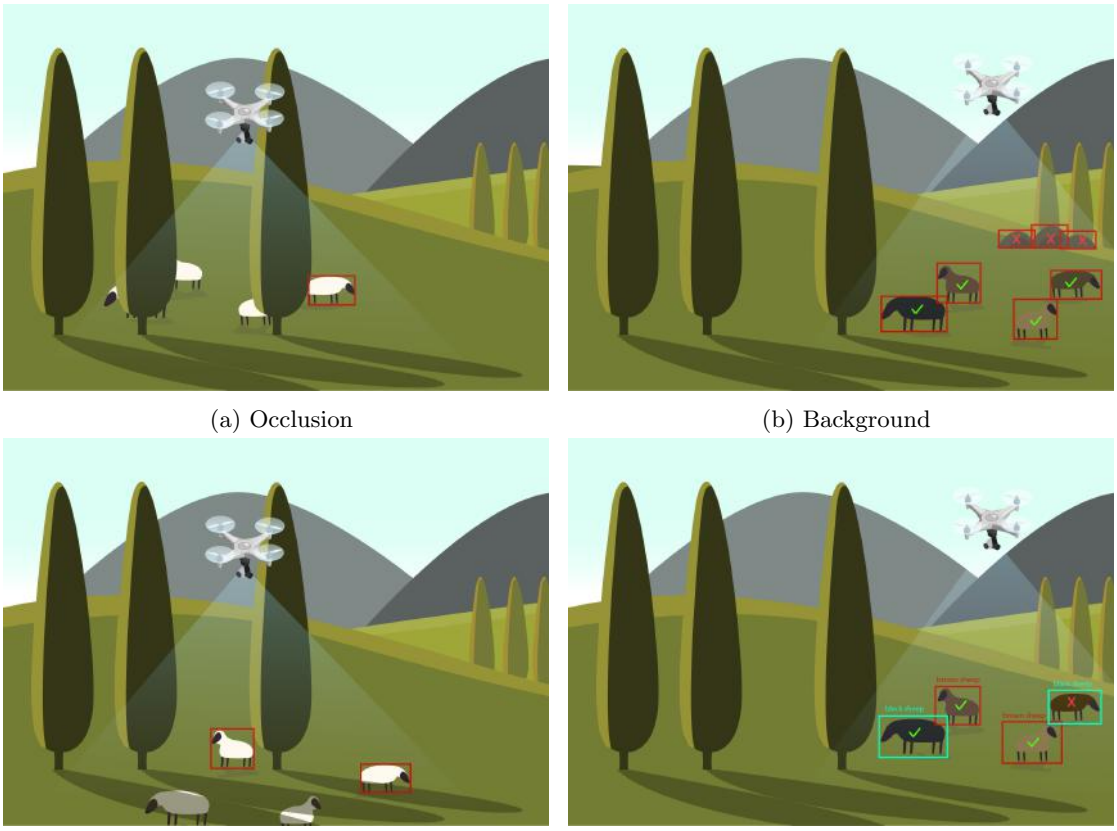
Once an object area gets identified through segmentation and feature extraction, it can be classified. Images are assigned to classes or object categories during the image classification step [32]. Different approaches to image classification exist, but in most cases, the assumption is that there are already labeled images for each class of objects. In these images, bounding boxes indicate which class of objects is present and its location in the image[3].

Classification involves comparing the collected information about the object area to be classified with the properties of the labeled images to determine the appropriate class. This information can be in the form of features of the object. Take the simplified example of a sheep. In the labeled images, horses often have the property of being large and four-legged, whereas birds have two legs and feathers and are, in comparison, small. Thus, if a feature corresponds to four legs in the images, it is more likely that a sheep is in the object area than a bird. Consequently, the ANN would classify this area as a sheep[3, 32].

The classification problem is about deciding whether or not a complex image shows at least one instance of a particular object category. The size of the object and its position in the image do not have to be determined. It differs from the localization problem, where all object instances must determine the position and size. Usually, this involves the specification of bounding boxes. If there are multiple object categories, the names of the instances must also be determined. This work considers the localization problem. As a sub-problem of the localization problem, the classification problem considers only one of many possible image sections[3].

Four common problems that can occur during the various steps of object detection are:

- **Occlusion:** Objects within the image may obscure each other. While the object may not be difficult for humans to recognize, certain object features can be hidden, which are important for recognition in an ANN.
- **Background:** The AI may falsely recognize objects in the background that should not be recognized.
- **Intraclass distance:** Specific image features in images of the same objects, such as different rotations, scales, color schemes, or other dependencies, often make the objects appear very different to the AI. For example, a sheep in high light conditions, such as sunlight, may look different to an AI than a sheep in dim light at dusk.
- **Interclass distance:** In the interclass distance, certain objects of different classes look visually similar, which makes it difficult for the AI to distinguish between the two objects.



(a) Occlusion

(b) Background

(c) Intraclass Distance

(d) Interclass Distance

Figure 3.1: Common issues encountered in object detection

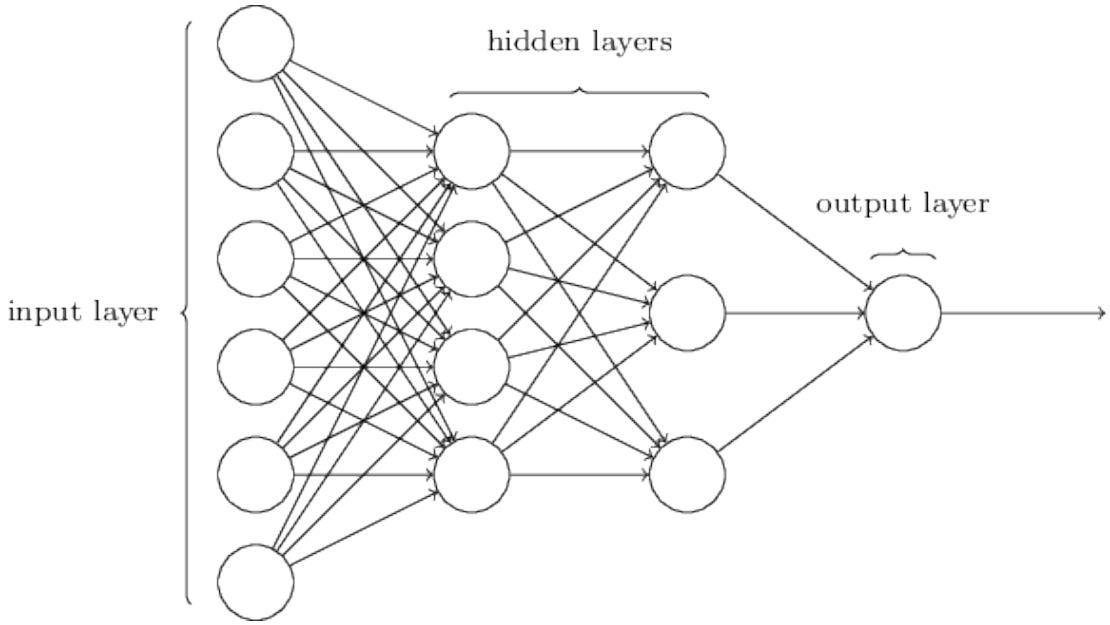


Figure 3.2: The layers of a neural network. Image source: [33].

3.2 Artificial Neural Networks

Artificial neural networks (ANNs) attempt to create a model for ML inspired by the knowledge gained in neuroscience research about the interaction of neurons and synapses[3].

Layers

A layer in an ANN is a set of neutrons at the same depth in the network. Traditionally the layers have been grouped into three categories based on their function, the input layer, the output layer, and the hidden layers. The input layer is the first layer responsible for feeding information into the network. The output layer is responsible for outputting a prediction based on what it receives from the previous layers. The input and output layers are connected by a number of intermediate layers of neurons. The more complex the task, the more neurons are connected. These intermediate layers are called hidden layers. The hidden layers use a large amount of training data for the learning process. From this information, simple patterns and structures are first extracted to form increasingly complex features that help solve increasingly complex tasks [3]. The final layer consists of the output neurons representing all possible outcomes.

Neurons

A single cell in the network is a neuron or perceptron unit. Its component and functions can be seen in figure 3.3. The neuron consists of several connections to the neurons in the previous layers, a body, where all the input signal and a bias term get added, and an activation function where the neuron's output to the next layer is determined[3].

Weights and biases

In a biological brain, not all neurons influence each other equally. How close the synapse is to the cell body influences the response strength. ANNs reproduce this by giving each connection between two neurons an individual weight. The higher the weight, the greater the signal one neuron exerts on another. The weights, as seen in figure 3.3, the strength of each connection between neurons,

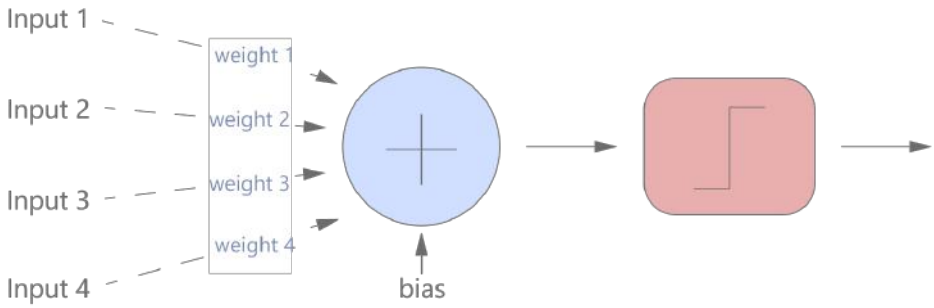


Figure 3.3: Perceptron unit. Image source [3]

are determined by the weights. A bias is a constant term added to the sum of signals from the other neurons. This parameter is also trainable but is specific to the cell body, and the weight of a connection does not determine its strength[3].

Activation function

How a neuron reacts to the signal can be determined using a mathematical function called an activation function. In the simplest case, a binary threshold function can be used. This function only has two levels of activity, on and off. If the incoming signals exceed the individual threshold value of the neuron, the neuron outputs one, and it fires. If the sum of signals remains below the individual threshold value, the neuron outputs zero. The disadvantage of such a function is that small changes in threshold can significantly affect the ANN. Continuous activation functions are used to avoid sudden increases in the activation level [3].

Earlier networks usually used sigmoidal functions. In newer networks, variants of Rectified Linear Unit (ReLU) functions are commonly used [3]. In the final layer of the network, a probability function is applied. This function converts its input to a class likelihood score based on the strength. The most commonly used function is the Softmax function (3.1) [3]

$$p_k = p(C_k|x) = \frac{p(x|C_k)p(C_k)}{\sum_j p(x|C_j)p(C_j)} = \frac{e^{l_k}}{\sum_j e^{l_j}}. \tag{3.1}$$

Loss function

The loss function is crucial for the way a neural network learns. The loss function calculates how well the model made predictions compared to the ground truth values of the dataset. The loss must be minimized over the training examples to optimize the weights [3]. For example, with the most commonly used Softmax activation function in the final layer, the loss function used is cross-entropy loss. Here as given by Nielsen 3.2 [33],

$$C = -\frac{1}{n}\sum_x \left[y \ln a + (1-y)\ln(1-a)\right]. \tag{3.2}$$

Back propagation

Backpropagation is the process where the network updates the weights and learns. The weights are updated using gradient descent by calculating the partial derivatives or gradients of the loss function. Then, using the chain rule, the gradient contribution of the neurons in the network is

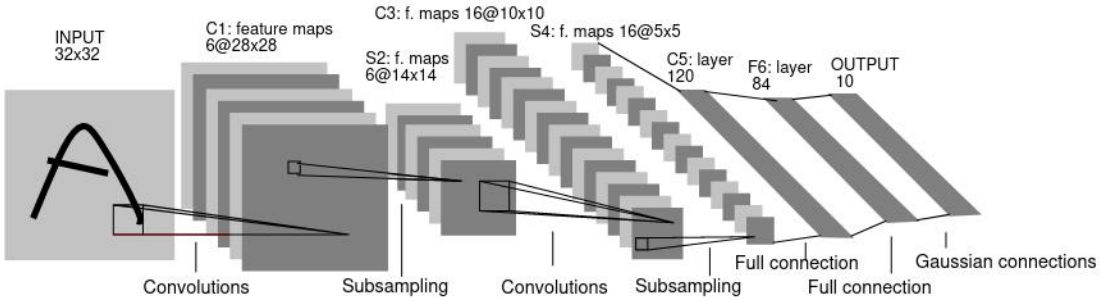


Figure 3.4: The CNN architecture for digit recognition described in LeCuns 1998 paper [34]

determined. From there, it is possible to work backward and strengthen the connections between neurons that contributed to finding the correct answer while weakening the ones that contributed to a wrong answer [3].

3.3 Deep Neural Network Architecture

3.3.1 Convolutional Neural Networks

A convolutional neural network (CNN) uses parameter sharing to reduce the number of weights used in the network. Unlike a fully connected network where all cells in one layer are connected to all cells in the next layer, the convolutional network organizes each layer into feature maps, using the mathematical operation convolution [3].

The first CNNs were developed by LeCun et al. in the 1990s and used ATNT to read checks automatically [34]. The network takes inspiration from the human brain's visual cortex, responsible for object recognition. CNNs are similarly used for machine vision and image classification. CNNs consist of multiple layers and extract features from raw data when processing two-dimensional images. In the process, low-level features (such as different types of edges) are extracted in the first layers. These combine in the deeper layers to form high-level features, such as object shapes. There is a distinction between convolutional layers and pooling layers [3].

CNNs [28] represent a particular architectural variant of neural networks that plays a central role in image recognition. Each input image is represented by its pixels as a matrix of dimension $height \times width$. The matrix contains a pixel value for each cell that reflects the characteristics of the pixel. For example, for a grayscale image, the pixel value corresponds to the irradiance and takes values between 0 and 255, black to white. In the case of color images, such as RGB images, the pixel value represents the intensity of a color in the image. As a result, each color has its color channel, representing an independent matrix. The input matrices are divided into small areas and sampled by the so-called convolution functions, allowing essential features to be extracted [3].

Convolutional Layers

In the Neural Network, several such filter kernels are arranged one after the other in several convolutional layers to extract increasingly abstract features and make them available to the following processing layers as a kind of feature map. The first layers extract simple features such as corners, edges, or curves. In contrast, the deeper layers often represent more complex shapes, such as outlines or specific features of an object [3].

The convolution process can be seen in figure 3.5. It works by sliding a kernel, often quadratic of size $k \times k$, over the image moved over the image in strides of length S. For each step, the pixels covered by the kernel are convolved by it. This resulting output value from this process over the

whole image is called an activation map. Depending on the kernel size and stride of which the kernel is moved, the resolution of the input image will be decreased. The edges of the input are often padded with P pixels to make sure that the down-scaling happens at a certain ratio [3].

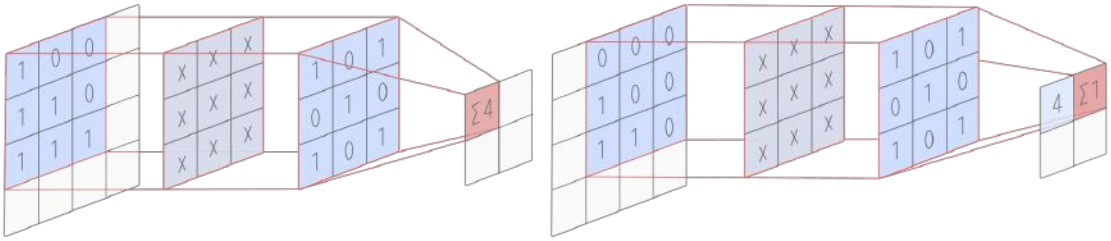


Figure 3.5: Convolutional function: A convolution being performed on a 4×4 image, using kernel of size $K = 3 \times 3$ and a stride of $S = 1$. Since no padding is applied to the image, the resolution of the resulting activation map will be 2×2, half the resolution of the original image.

Pooling

For many high-dimensional feature maps, there is a need to reduce the complexity between the convolutional layers by using a process known as pooling. This process is applied in an intermediate layer between two convolutional layers [3].

A pooling layer reduces the input resolution by compressing the large amounts of data received from the convolutional layers. The layer uses a kernel with an area of $N \times N$ values as input and outputs a single value. For example, a 2x2 Max pooling layer would halve the input resolution. Different pooling layers exist, but the most commonly used is max pooling, where the pooling layer outputs the maximum value of its input. Another type of pooling layer is average pooling, where the kernel outputs the average input values [3].

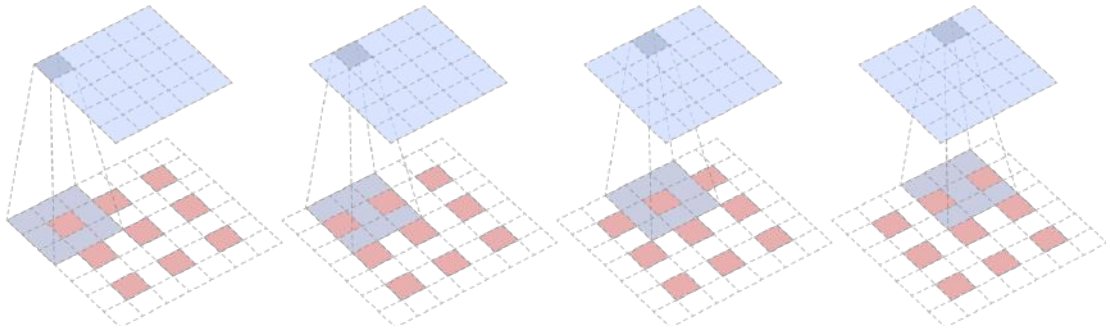


Figure 3.6: Pooling.

Network Hierarchy

The first part of a CNN deals with the calculation of image information. This part of a CNN is also known as the Feature Extractor since it is responsible for extracting representative features to describe an image. After this, the second part works like a traditional classification algorithm, in which the previously extracted features serve as input variables for a simple classification function. At this point, a fully-connected layer ensures the reassembly of the processed information from the previous layers. The number of neurons in this layer usually corresponds to the number of classes between which the network is to distinguish [28]. The structure of a CNN as proposed by LeCun et.al in 1998 [34] is showed in 3.4. In addition, CNNs have different scanning functions and network depths depending on the architecture variant [3].

3.4 Model evaluation

How well a trained system performs is reflected in the values "Precision" and "Recall," which find use in the creation of a Confusion Matrix 3.7. This matrix maps four values:

- True Positives (TP): Class correctly recognized.
- False Positives (FP): Class incorrectly recognized
- True Negatives (TN): Class correctly not found
- False Negatives (FN): Class incorrectly not found

With these data, the values for Precision and Recall can be calculated according to the following formulas.

$$Precision = \frac{TP}{TP + FP} \tag{3.3}$$

Precision is the proportion of correct detections of a class in context to this class's total number of detections.

$$Recall = \frac{TP}{TP + FN} \tag{3.4}$$

The recall is the proportion of correct detections of a class related to the total occurrence of a class, i.e., including ground truth.

	Predicted Positive (1)	Predicted Negative (0)
Actually Positive (1)	True Positive (TP)	False Negative (FN)
Actually Negative (0)	False Positive (FP)	True Negative (TN)

Figure 3.7: Confusion Matrix

These values form the basis for other metrics, such as average precision (AP).

Intersection Over Union (IoU, Jaccard-Index)

A common method to assess how an object has been recognized in images is the IoU or Jaccard index. The IoU is calculated on an image-by-image basis according to the following formula for each class [3].

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}} \tag{3.5}$$

The overlap area refers to where the bounding box detected by the model overlaps with the ground truth. The Area of Union defines the combination of the bounding box areas and the ground truth. The IoU is calculated per class. Furthermore, the average IoU can be specified for each image by calculating the average value of the IoU of all classes in the image. A value of 0; 5 or more is considered good recognition [3].

Average Precision and mean Average Precision

Average Precision (AP) values are a metric in object detection. When calculating the AP, the IoU is used as a threshold value. There are various procedures for determining the threshold value for calculating the AP. Since an IoU of 0.5 or more is considered a good detection, detections with an $IoU >= 0.5$ are assumed to be correct. AP values with this threshold are specified as AP_{50}. A precision-recall curve $p(r)$ is generated with this threshold. The AP corresponds to the area under this curve [3].

$$AP = \int_0^1 p(r)dr \tag{3.6}$$

The use of this metric is usually when precise localization is not required. Considering the IoU, this metric cannot distinguish between a precise and an inaccurate model. The calculations are done class by class when applied to an entire dataset. The mean Average Precision (mAP) is calculated as an average value over an IoU threshold that increases in 5% increments from 50% to 95% [3].

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{3.7}$$

3.5 Influential CNN types

3.5.1 Region-Based Convolutional Neural Networks

Several other network variants have emerged based on the CNN architecture in recent years. These include the approach of Region-Based Convolutional Neural Networks (RCNNs), in which classical CNNs are combined with Region Proposal Networks [35]. The network enables the localization of objects in addition to classification. First, image regions that stand out from the background due to their image structure are determined. Then features of these image regions are extracted through CNN and used for object determination. Based on this, regions with the same classification are used to determine the positions of these objects. A computationally efficient implementation of this concept is the Faster-R-CNN architecture [36]. The latest iteration of the architecture Mask R-CNN [37] is now widely used. As seen in figure 3.8, object recognition by R-CNN works in three steps [35]. In the first step, a selective search is performed on the input image to identify areas that may contain objects. These areas have the name "region proposals." The second step is to adjust the size of the region proposals to a CNN. This CNN extracts features and classifies objects according to the features found. Finally, bounding boxes are calculated using a support vector machine trained with CNN features with the region proposals.

3.5.2 YOLO - You Only Look Once

The object detection method YOLO was published in 2015 by Redmon et al. [38] and has since been further developed in several steps. The authors consider object detection a regression problem in which objects are identified from the input image pixels with the help of a neural network. In this process, a recognized object is described by a bounding box that unambiguously determines its position and size. In addition, for each detected object, the number of class probabilities equals the number of possible classes in the respective application. A class probability value gets determined for each class - their sum equals one. A detected object becomes assigned to the class with the highest class probability. One of the central goals of YOLO is real-time object recognition. Due to its properties as a one-stage object detector, only one neural network uses the entire object recognition process run through per image. In the process, the image is only looked at once (You

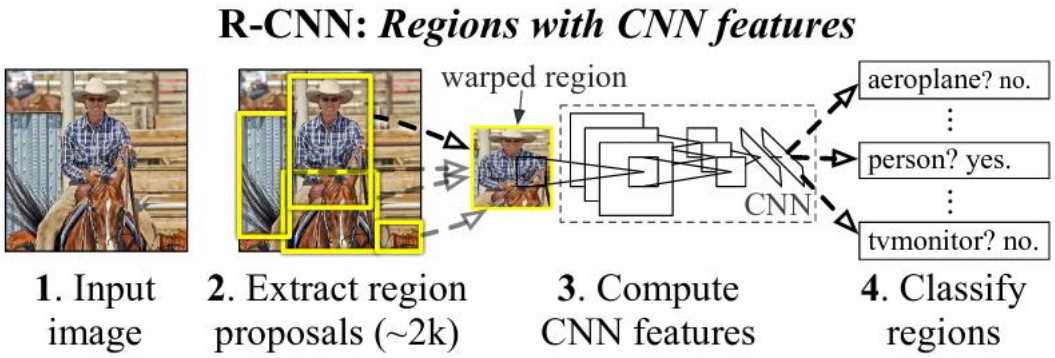


Figure 3.8: How R-CNNs work from Girshick et al. 2013 [35]

Only Look Once). As a result, YOLO can be optimized end-to-end according to its performance in object detection. The earlier mentioned two-stage object detectors do this in two operations. This group of networks includes the R-CNN [35], Fast RCNN [39], and Faster RCNN [36].

Simply put, two-stage methods create region propositions in the first step. Region propositions are suggestions for areas in which objects could be located. In the second step, these regions are used, and the content is considered a classification problem. That means that the class of the mapped object is determined. Redmon et al. describe that these object recognition systems cannot deliver real-time performance, whereas YOLO is very performant due to its design [38].

YOLO divides the input image into a $S \times S$ grid. The cell that contains the center of an object on the image is "responsible" for recognizing the object - thus, ideally, each object is determined by precisely one cell. B bounding boxes belong to each of these cells. The YOLO object recognition model design arbitrarily chooses the number of cells S and the bounding boxes B. For example, Redmon et al. use a grid of 7×7 cells with two boxes each in their model [38].

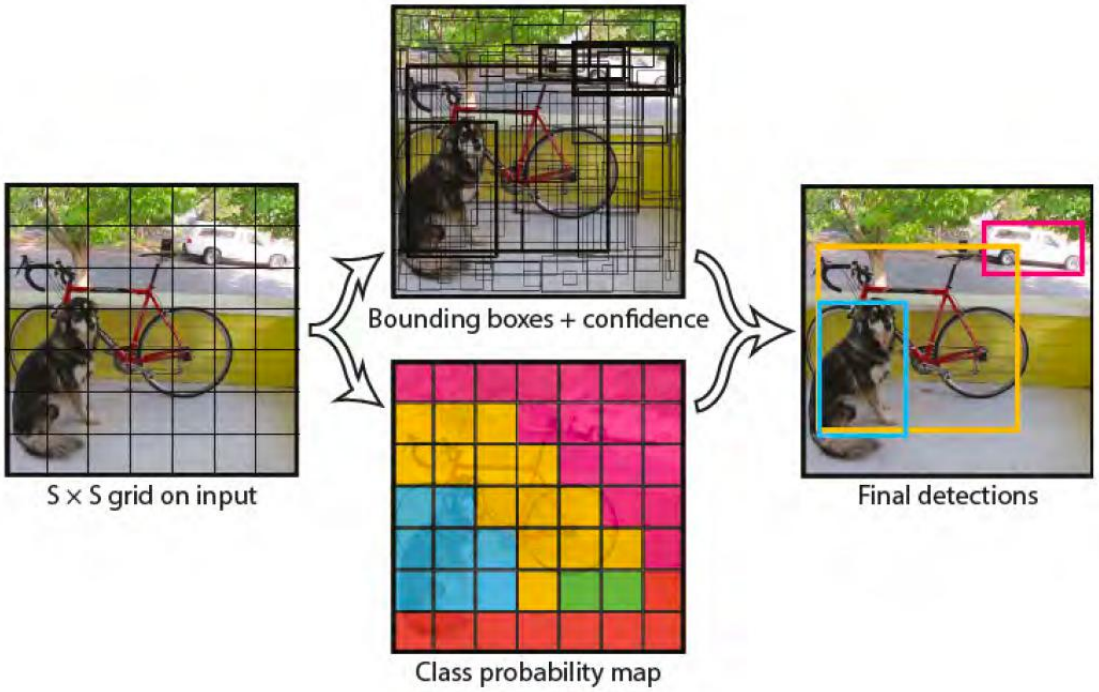


Figure 3.9: YOLO Object Detector [38]

A bounding box is described by coordinates $(x; y; w; h)$ and a confidence score [3]. The $(x; y)$

coordinates define the center of the bounding box as a relative offset to the cell to which they belong. The $w; h$ coordinates describe the height and width of the box relative to the whole image. Bounding boxes define the position and size of objects in images. The confidence value is the last value that completely describes a bounding box. That indicates whether the respective bounding box contains an object and how well it coincides with this object. The confidence score is defined as

$$Confidence = Pr(Object) \times IoU_{pred}^{truth} \tag{3.8}$$

The objectness score, $Pr(Object)$, determines how likely the box is to contain an object. IoU defines how accurate the dimensions and position of the box are compared to the dimensions of the contained object. The confidence scores should be zero when there is no object in the cell. In cases where an object has been found, the confidence score should ideally be the same as the IoU of the predicted box and the ground truth box of the object. [38]

YOLOv2

With the publication of YOLO, Redmon et al. continued to work on improving their object detector. They discovered that many localization errors occur compared to two-stage approaches such as Fast-RCNN or Faster-RCNN [40]. They also claim that the recall is relatively low in comparison. To address this, Redmon et al. published the paper "YOLO9000: Better Faster Stronger" (later referred to as YOLOv2) [40], an improved version of YOLO that focuses on fixing the problems mentioned above. The intention was to maintain YOLO's high speed and quality of recognition. The authors experimented with various techniques to improve, introducing anchor boxes and batch normalization to the network architecture, which have proven useful in neural networks, especially object recognition. Another important change to decrease localization errors was allowing each grid cell to output five bounding box detections instead of a single prediction per grid cell as in the first version. [40]

YOLOv3

Redmon et al. published their last improvements for YOLO, YOLOv3, in 2018 [41]. Several changes were made in YOLOv3 that increased the size of the network but significantly improved the results. YOLOv3 used feature pyramids, Darknet39, and class prediction.

The main changes from YOLOv1 and YOLOv2 are in the architecture. The original YOLOv1 and YOLOv2 output a single prediction for bounding boxes with a 7x7 and 13x13 cell grid. However, this works well for objects with large bounding boxes, but the performance is worse for smaller objects with smaller bounding boxes. YOLOv3 uses three different grid sizes that output predictions for bounding boxes at three locations in the network. The different sizes are a 13x13 grid, a 26x26 grid, and a 52x52 grid. That way, the network is better suited to detect small objects. As a result, the number of predictions from YOLOv1 and YOLOv2 also increases significantly.

Version	Box Number Calculation		Result
YOLOv1	7×7	=	49
YOLOv2	$13 \times 13 \times 5$	=	845
YOLOv3	$13 \times 13 \times 3 + 26 \times 26 \times 3 + 52 \times 52 \times 3$	=	10647

Table 3.1: Increase in output bounding boxes between YOLOv1, YOLOv2, and YOLOv3.

Due to concerns about using his findings for military applications and the impacts it could have on privacy, Redmon announced in February 2020 that he would stop his research in the field of computer vision [42].

YOLOv4

Following Redmon's departure from YOLO research, Bochkovskiy et al. published a modified version of YOLOv3 called YOLOv4 in April 2020 [43]. As part of this version, a Cross Stage Partial Network (CSPNet) [44] is used in Darknet, forming a new feature extractor backbone called CSPDarknet53. The basis of the convolutional architecture is a modified DenseNet [45], which uses a dense block to transfer a copy of the feature map from the base layer to the next layer. DenseNet has, among other advantages, fewer problems with gradient vanishing, better backpropagation, eliminating computational bottlenecks, and improved learning. The Neck system is based on the SPP (Spatial Pyramid Pooling) layer and Path Aggregation Network (PANet). These two layers are used for feature aggregation, which improves the receptive field and filters out essential features from the backbone. The header is also made up of a YOLO layer. In the first step, the image is fed into CSPDarknet53 for feature extraction and then into the PANet for fusion. Finally, the YOLO layer generates the results, similar to YOLOv3 [43].

In YOLOv4, the authors divide their used methods into "Bag of Freebies" and "Bag of Specials." The Bag of Freebies group methods improve the quality of object detection and possibly increase training time but have no impact on inference time. Such methods include Complete IOU Loss (CIOU), Drop Block Regularization, and various augmentation techniques. On the other hand, Bag of Specials groups methods increase inference time but can significantly improve the quality of the object recognition model. These include mish activation, Diou- (Non-Maximum-Suppression) [46] and modified path aggregation networks.

YOLOv5

Glenn Jocher from Ultralytics developed YOLOv5 [47]. The project is an adaptation of the YOLOv4 network implemented in the PyTorch framework. While there is still no paper out from the model creators, research by Nepa and Eslamiat from 2022 finds its performance compared to its predecessor YOLOv4, outperforming it on accuracy while matching it in speed [5]. Jocher is credited as the creator of the Mosaic image augmentation technique used in YOLOv4 [43]. This augmentation method is used for all images by default when training the YOLOv4 models [47]. During training, the YOLOv5 uses data augmentation to improve generalization and avoid over-fitting. The framework applies online augmentations of the image space and the colorspace when the image is being loaded for training. The chosen image is put together in a "Mosaic" with up to three randomly selected images each time it is loaded for training. This means that an image will never be presented in the same way. This image augmentation is not done when loading images for the validation set [47]. The architecture of YOLOv5 can be seen in figure 3.10.

3.6 Potential problems when training the network

3.6.1 Over and underfitting

An ML model is expected to function in a generalized way. In the case of object recognition, this means that the model finds objects in data that it has not seen before. However, performance can deviate from the desired generalization in two ways. These cases are called "overfitting" and "underfitting" [3].

Overfitting occurs when the model learns too many features from the training data. In other words, the model is not generalized enough to recognize objects in the new data [49]. A typical case is that a model has been trained for too long and over-fits the training data. It is similar to memorizing where objects are in the training images instead of learning how to detect them. As a result, noise in many noisy images may be recognized as a feature, and the model may not recognize objects that are not noisy. Overfitting occurs more quickly when training with a small data set [3].

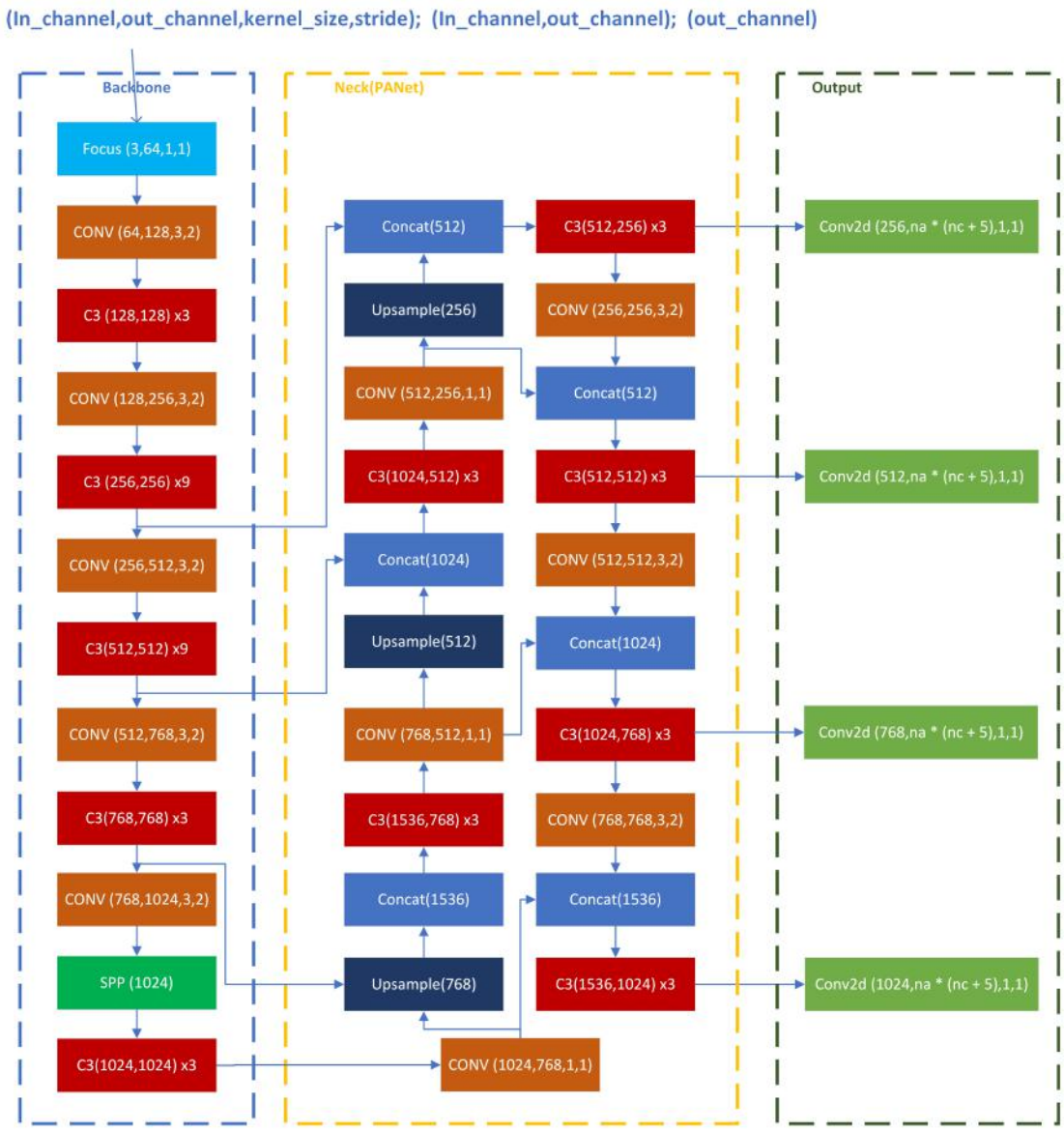


Figure 3.10: The YOLOv5 architecture. Image source: [48].

Underfitting occurs when the model has not been sufficiently trained. For example, by too short training or too little training data. In this case, the model will not have learned enough features to recognize variations of objects outside the training set [3, 50].

3.7 Dataset Composition

The composition of the dataset has a significant impact on the success of the network in learning. Therefore, a well-composed dataset should meet specific requirements related to dataset shift and dataset bias[3].

Dataset bias occurs when a dataset has more representatives of some classes than others. As a result, the dataset has a majority and minority classes. This distribution leads to the network model getting more reinforcement from the majority class and less from the minority class [51].

Torral describes dataset shift as a problem where the dataset that the training set and the validation set are too different from the test set [51]. A cause of this can be that the training and validation

data are not captured in the same domain or fashion as the test set or where a trained network model is being applied. When the dataset shift is not identified, training the model will be better than deploying the model in a test setting or production [51].

Chapter 4

Experiment

This section outlines the methods used to perform the experiment on which this thesis is built. Part of this is detailing the tools and frameworks used in the project. This concerns how the dataset was acquired, how it was labeled, and how it was pre-processed before starting training on the machine learning framework.

4.1 Requirements

The following requirements were considered necessary when selecting the tool for the experiment. These include the choice of the correct machine learning framework, which is crucial for the quality of object recognition. In this area, various methods and libraries must be tested for suitability and selected accordingly. In addition, requirement-specific training data is necessary to train the system.

Machine learning model

Once the training is complete, the machine learning model should be in a format compatible with the chosen framework. The input values should only require the input of an image. The output values need to contain the parameters of a bounding box for each class and an associated confidence value. Due to the possible deployment on lightweight systems, such as mobile devices, and possible post-deployment training, the model's size and training time will also be considered.

Training platform

Since standard object detectors based on CNNs have the advantage that the calculations required to train them can run in parallel, this should be taken advantage of using powerful Graphics Processing Units (GPUs), commonly referred to as graphic cards. Because of their architecture with several thousand cores, GPUs offer a significant performance boost over CPUs when training CNNs. In addition, some manufacturers offer graphics cards with exceptional support and optimization for machine learning algorithms, such as the Nvidia Tesla series [52].

Training data

In order to train a machine learning model, relevant data must be available. This dataset will be used to train the model. For object detection, this information must be produced as labeled image data. It is not only about the amount of data but about using data that shows the object with as much variation as possible. In this way, the model learns to recognize objects in different

contexts and conditions. For this purpose, image data must be available that has been created under different illumination conditions and viewing angles.

Additional data variability must be artificially created by augmenters using images of the object with different exposures. Then, a dataset large enough for dividing into training and validation data is created for training. Then, additional image data is generated using data augmentation techniques. If the training data is sufficient is determined by evaluating it with metrics such as mAP.

Labeling

Since captioning would be very time-consuming without a dedicated tool, it should be able to import images, create captions by dragging them with the mouse, and then automatically save them for each image. In addition, web applications that allow images to be uploaded to cloud storage could be used for collaboration.

The labeling software should support different formats when importing and exporting labels to make the dataset more accessible to other students. Optional functions include editing images and adding augmentations, such as cropping, flipping, rotation, and mirroring.

With the aim of fast labeling more significant amounts of data, it should also be possible to pre-label the data with already trained networks so that only manual corrections have to be made. This way, the work will minimize the time needed to generate large data sets. For example, the initial training can be performed with only a few images. The result can then be used to pre-label more images and use them again for training.

4.1.1 Research Questions

Baseala on the dataset from Johansen's thesis from 2020 [12] and experiences of collecting new images to expand the data set, it became apparent that the low resolution of the IR image sensor on the drone would lead to data of varying quality. Therefore, this was taken to examine how much impact the low-resolution data would have when combined with the high-resolution data from the RGB image sensor. Based on this, the following research questions were formulated:

- **RQ1:** Does training on IR images improve model performance overall?
- **RQ2:** Does training on IR images help find sheep occluded by brush and vegetation in forested areas?
- **RQ3:** Is the extra cost of an IR image sensor worth it for a farmer?

4.2 Data collection

4.2.1 Equipment

The project was conducted with a DJI Mavic 2 Enterprise Dual (M2ED) drone. This camera drone is marketed for professional use for plant inspections and search and rescue missions. The advantage of the M2ED is that it has a built-in camera unit consisting of an RGB camera combined with a Flir thermal imaging camera. The thermal imaging camera is permanently mounted and stabilized with the help of the 3-axis gimbal. Previous flight devices have usually operated without stabilization and only attached a thermal imaging camera to the flight device through a rigid connection. The M2ED thermal camera features a sensor resolution of 160×120 pixels and a maximum image size of 640×480 (4:3). In video mode, the thermal camera technical specifications are 640×360 pixels and a frame rate of 8.7 frames per second [27].

DJI Mavic 2 Enterprise Dual	
Flight Time	31 min
Speed	72 kph
Range	8000 km
Internal Storage	Micro SD, 24GB Onboard
Camera	12 MP
Sensor	Sony 1/2.3
Lens	f/2.8-f/3.8
Zoom	2x Digital 3x Electronic
Shutter Speed	Electronic Shutter 8-1/8000s
ISO Range	Video: 100-3200 Photo:100-3200 (manual)
Color Mode	D-Cinelike
Max Bitrate	100 Mbps
Battery	3850 mAh (heated)
Transmission System	OcuSync 2.0

Table 4.1: The specification of DJI Mavic 2 Enterprise Dual.

Figure 4.1: The DJI Mavic 2 Enterprise Dual. Image source [53].

4.2.2 Collection approach

The drone was flown in multiple sessions, with the duration of the sessions depending on the battery life. For the flights, three batteries were used, each lasting about 20 minutes, limiting each session to approximately one hour.

The drone was positioned straight above the sheep, and images were taken at 40, 50, and 60 meters altitudes. For pictures taken in areas with denser vegetation, attempts were made to capture the sheep while partially or fully covered by foliage or other objects.

4.2.3 Locations

Data collection was conducted during the period from September to November 2021. These images were taken at three locations under different terrain and weather conditions. In total, 1216 image pairs were collected.

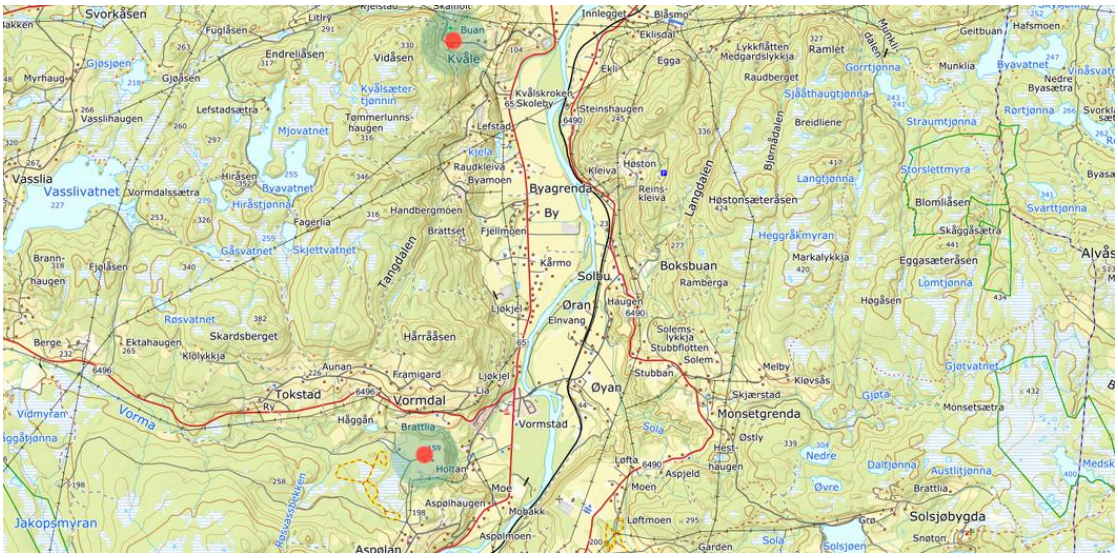


Figure 4.2: Map of Orkdal. The circles show were the sessions were conducted.

Holtan infield

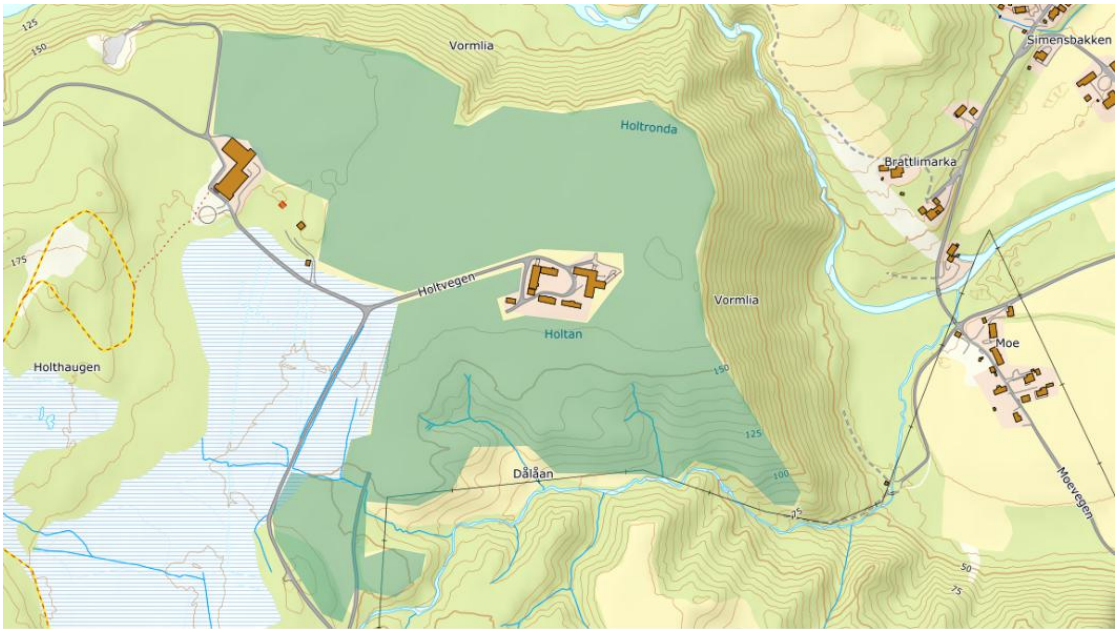


Figure 4.3: Map of Holtan infield. The marked area is where the sessions were conducted.

Sheep in an open field are the main part of the footage. Due to seasonal changes, there are variations in vegetation, degree of forestation, and background color. Most sheep were Norwegian Dalasau with white wool. In addition, there was a smaller proportion of Norwegian Spælsau with a greater variety of wool colors.

Holtan outfield

The footage from the outfield of the Holtan farm was taken in a forested area consisting of felling fields with varying degrees of regrowth, forest clearings, and dense spruce and deciduous forest. The data collection occurred in late fall when most sheep had already moved from the outfield into the farm's infield areas. Given the data gathering time, the sheep in the outfield proved challenging

(a) Dalasau (b) Spælsau

Figure 4.4: Two different breeds of sheep. Image source:[54].

Figure 4.5: Selection of images from Holtan infield. The higher surface temperature of grazing cows compared to sheep can be seen in the left most image. The images shows the variation of environments in this area, with an open field, a hillside, and the forest edge

to locate, and most footage was captured in three separate sessions. These sessions were conducted at dusk and colder temperatures.

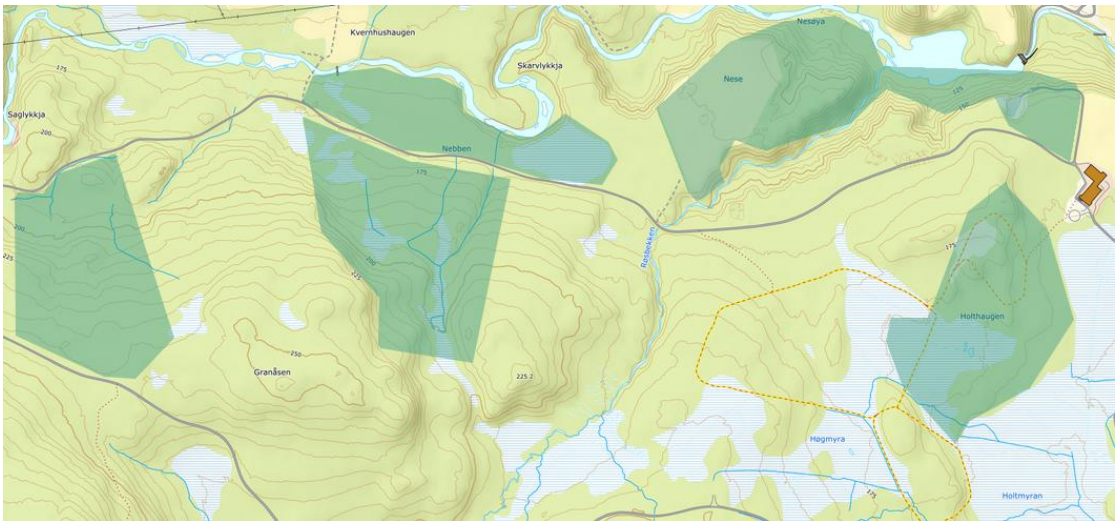


Figure 4.6: Map of Holtan outfield with a rough outline of where the sessions were conducted.

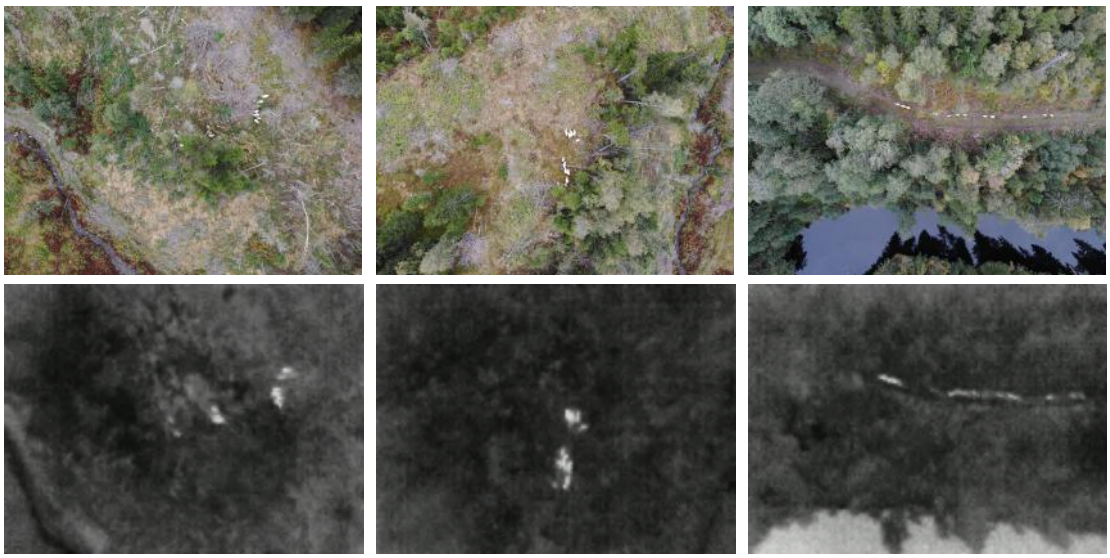


Figure 4.7: Selection of images from Holtan outfield.

Buan Farm outfield

The Buan Farm footage was taken midday in the farm's outfield. The area in which the sheep were walking was a marsh area in the middle of a felling field, with sparse regrowth. The images were taken on a sunny day which led to some shadows being present in the images. The sheep in the footage are old Norwegian spælsau, which have a more significant representation of multi-colored sheep.

As can be seen in figure 4.9, the sheep are less visible in the images compared to the sheep in the open field from Holtan infield 4.5, and the images from Holtan outfield 4.7 that was taken on a cold evening.

It can be hard to differentiate between sheep and the tree stumps remaining from the felling in the IR images. It can also be seen that the intense sunlight significantly impacts the IR images and that the areas the shadows convey little information compared to the sunny areas.

Figure 4.8: Map of Buan outfield. The marked area gives an estimate of where the sessions were conducted.

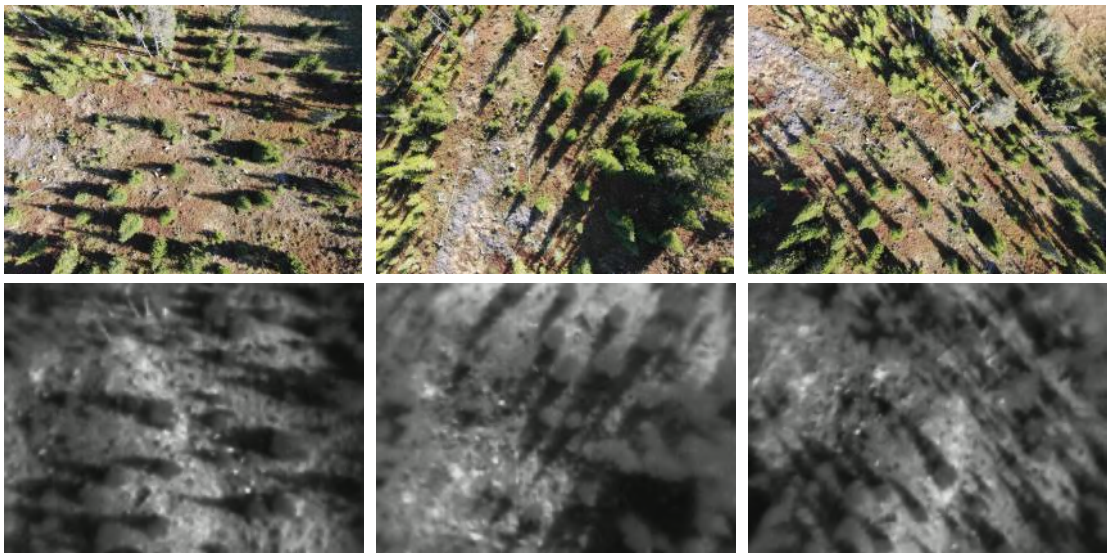


Figure 4.9: Selection of images from Buan outfield.

4.2.4 Dataset labeling

Based on the criteria listed in the requirements section, the choice fell on the Roboflow collaborative labeling tool [55]. The annotation was done In collaboration with two other student teams writing their master's thesis in sheep recognition. While Roboflow can suggest labels based on models trained on the datasets such as Common Objects in Context (COCO), the accuracy of these predictions was too low to be helpful in labeling, and it was easier to label the sheep from scratch. One reason could be the difference between the sheep images in the COCO dataset, i.e., images of sheep taken from the ground, and our dataset, which consists of aerial images.

The dataset was annotated with black, brown, gray, and white sheep classes. During the data collection phase, approximately 1300 image pairs were collected. Images of poor quality were discarded, leaving 1117 newly labeled image pairs.

4.2.5 Final dataset

In addition, Kari Meling Johansen shared the dataset that she had used for her thesis. Overall, 562 images from her dataset were used in the final dataset. The newly collected images with the preexisting dataset gave a final merged dataset consisting of 1679 images with the class distribution shown in table 4.2.

Color	Black sheep	Brown sheep	Grey sheep	White sheep	Total
Count	1806	913	2401	10433	15553

Table 4.2: The distribution of the type of sheep in all images.

4.3 Image Pre-Processing

4.3.1 Distortion correction

The dataset consists of IR and RGB images. In the IR images, significant distortion is observed at the edges of the images. Therefore, camera calibration was used to correct this to match the image pairs and estimate a transformation between the two image pairs.

Camera calibration software

According to Szeliski, radial distortion is when the pixels of an image are displaced proportionally to the radial distance from the image center. In the case of barrel distortion, pixels shifts towards the center of the image. In pincushion distortion, on the other hand, the pixels shifts away from the center [3].

In the combined images, where several straight lines of roofs cover the outer edges of the images, as shown in Figure 4.10, it is clear that the distortion in the IR images captured by the drone is very similar to the barrel-shaped distortion.

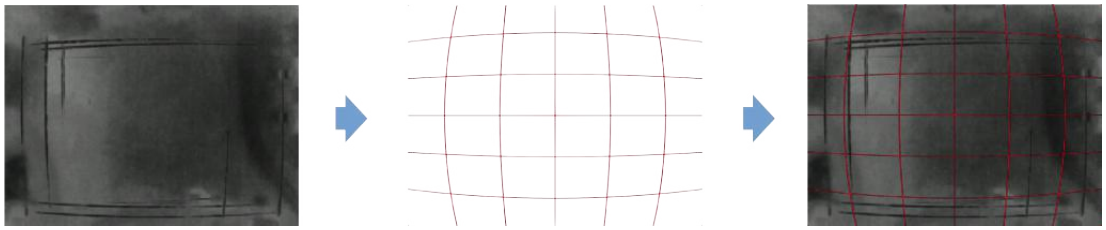


Figure 4.10: Comparison between an example for barrel distortion and a combined image where several straight lines from rooftops are overlain each other.

As part of her master's thesis, Johansen used an overhead projector and a transparent chessboard image to estimate the distortion correction coefficients used to correct the camera distortion in the infrared camera [12]. It required a large set of images with a specific grid pattern to work optimally. Using these images for calibration requires a great deal of manual effort to mark the intersection points of the grid. It is also considered tangential distortion, which is not observable in the images.

Inspired by this, the distortion correction process was attempted to simplified by manually adjusting the distortion coefficient parameters 4.2 using a modified version of the Python package VirtualCam [56] and a simple graphical user interface with two sliders controlling the coefficients. Scaling and aligning the images would then be done using the estimate transform method from the SkImage toolkit [57]. This method estimates a transformation from one image to another when given the coordinate pairs of corresponding points in the images.

To correct the distortion, a simplified pixel-wise radial distortion correction model as presented by Szeliski [3] can be used. This model is based on the Browns model for distortion correction but does not consider tangential distortion. For the position of a single pixel with coordinates in a homogeneous format, the function is written as follows:

$$\mathcal{D}(\bar{x}) = \begin{cases} \hat{x} & = x(1 + k_1 r^2 + k_2 r^4), \\ \hat{y} & = y(1 + k_1 r^2 + k_2 r^4), \\ \hat{w} & = 1 \end{cases} \tag{4.1}$$

where $r^2 = x^2 + y^2$ where x and y are the pixels coordinates relative to the image center. The other variables are the distortion coefficients,

$$Distortion\ coefficients = (k_1 \quad k_2), \tag{4.2}$$

which determines the strength of the displacement.

For shifting the coordinates back to the image space the distortion correction algorithm is dependent on the camera matrix

$$\mathbf{K} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

where f is the camera's focal length, and (c_x, c_y) is the center of the image. In this distortion correction model, the center of the image can be set as $c_x = W/2$ and $c_y = H/2$. Where W and H are the width and height of the image.

The process of distortion correction is as follows: the first two lists that are of length $k = n \times m$, the number of pixels in the image, are created. The two lists contain the respectively, the x and y index of the pixels. The list is normalized to the range $[-W/2, W/2]$ for x coordinates and $[-H/2, H/2]$ for the y coordinates. The list of coordinates is then merged together with two lists of the same length, one with a predetermined Z component and another with $w = 1$ components, to create homogeneous coordinates. This results in the following $4 \times k$ matrix:

$$\mathbf{P} = \begin{bmatrix} x_0 & x_1 & \dots & x_k \\ y_0 & y_1 & \dots & y_k \\ z_0 & z_1 & \dots & z_k \\ 1 & 1 & \dots & 1 \end{bmatrix}. \tag{4.4}$$

This matrix is then multiplied by the translation matrix

$$\begin{bmatrix} \mathbf{I} & \mathbf{t} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \end{bmatrix}, \tag{4.5}$$

which gives the transformation of the homogeneous coordinates to inhomogeneous coordinates

$$\bar{\mathbf{P}} = \begin{bmatrix} \mathbf{I} & \mathbf{t} \end{bmatrix} \mathbf{P} = \begin{bmatrix} x_0 + t_x & x_1 + t_x & \dots & x_k + t_x \\ y_0 + t_y & y_1 + t_y & \dots & y_k + t_y \\ z_0 + t_z & z_1 + t_z & \dots & z_k + t_z \end{bmatrix}. \tag{4.6}$$

Then column-wise perspective projection by dividing all elements with z is performed on the matrix as follows

$$\bar{x} = \mathcal{P}(p) = \begin{bmatrix} x/z \\ y/z \\ 1 \end{bmatrix}. \tag{4.7}$$

where $\bar{x}$ is a single pixel in 2D space, and $\mathbf{p}$ is a single pixel in 3D space.

Then each column in the matrix-vector was shifted using equation 4.1

$$x' = \mathcal{D}(\bar{x}) = \begin{bmatrix} x(1 + k_1 r^2 + k_2 r^4) \\ y(1 + k_1 r^2 + k_2 r^4) \\ 1 \end{bmatrix} = \begin{bmatrix} \hat{x} \\ \hat{y} \\ 1 \end{bmatrix}. \tag{4.8}$$

Resulting in a matrix of shifted coordinates

$$\mathbf{X}' = \begin{bmatrix} \hat{x}_0 & \hat{x}_1 & \dots & \hat{x}_k \\ \hat{y}_0 & \hat{y}_1 & \dots & \hat{y}_k \\ 1 & 1 & \dots & 1 \end{bmatrix}. \tag{4.9}$$

Finally, the normalized coordinates are projected back to the image space by multiplying with the camera matrix

$$\mathbf{K}\mathbf{X}' = \mathbf{M} \tag{4.10}$$

creating a matrix $\mathbf{M}$ where the first and the second row of the matrix are maps for x and y coordinates for the pixels in an image. After the maps were created, the OpenCV remap function was used to remap the pixels in the image, correcting the distortion.

In this experiment, the following parameters were used for the camera matrix

$$\mathbf{K} = \begin{bmatrix} -100 & 0 & W/2 \\ 0 & -100 & H/2 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.11}$$

the translation matrix

$$\begin{bmatrix} \mathbf{I} & \mathbf{t} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -85 \end{bmatrix}, \tag{4.12}$$

The distortion coefficients were set in a simple program with a graphical user interface (GUI) which was developed to calibrate the images. The program allows the user to open a window where they can use sliders to configure the k_1 and k_2 coefficients and get visual feedback on how changes affect the image. This window is shown in 4.11.

After suitable coefficients were found, an affine transformation was generated by marking points identified in both images, mapping the infrared image to the corresponding coordinates in the

Figure 4.11: Sliders to used to control the K variables used for distortion correction. A GUI program using a slider to correct the distortion and the estimate_transform method from the Skimage toolkit to estimate the transformation between the two images.

color image. An image with a similar resolution is thus created, and the applied transformation straightens the distorted lines.

Due to the smaller field of view of the IR images, the image pairs were cropped to the size of the maximum 4:3 sized rectangle that could fit within the IR image. As a result, the resolution was 3200×2400.

This approach made it possible to obtain results comparable to those obtained with the camera calibration method used in the earlier work of Johansen [12].

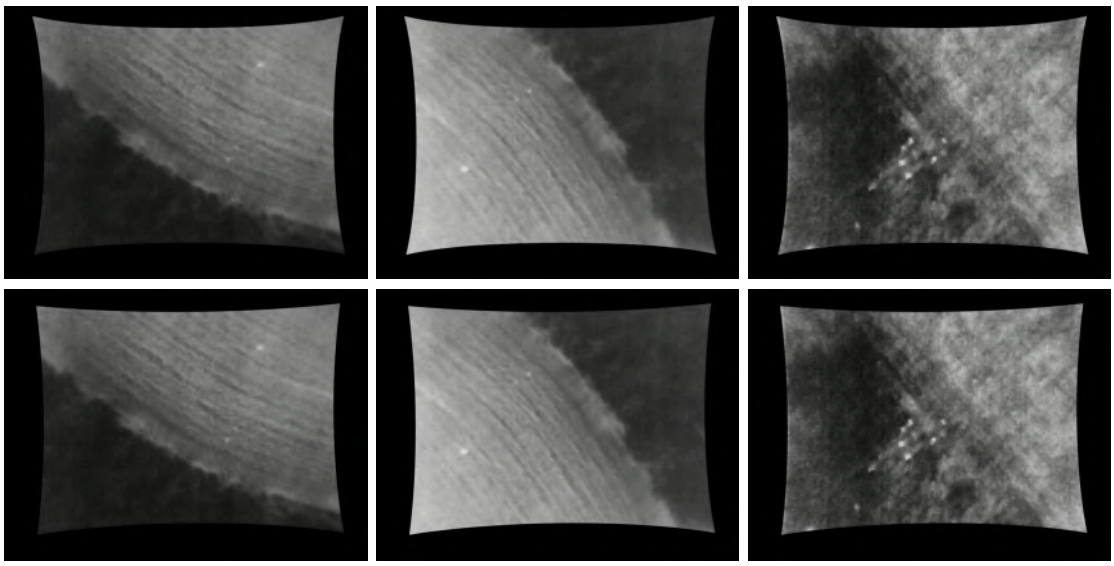


Figure 4.12: Comparison between the simplified UI image undistortion to the chessboard calibration done in previous thesis. The top row of images shows the UI calibrated images, while the bottom row shows the images undistorted used chessboard calibration.

4.3.2 Combining RGB and IR images through color space shift

The IR image is grayscale in the color range of 0 to 255, where 0 is black and 255 is white. In addition, a threshold value of 50 was set to remove noise from the images, where pixels with an image value under the threshold were defined as black.

```
if p > 50 :
        p = p
else:
        p = 0
```

The images were converted to the LAB color space and split into three L-A-B channels. Then, the result of the thresholded IR images was added to the A channel of the images as follows:

```
a = a*0.5 + IR*0.5
```

These channels were then combined again and converted to the RGB color space. The resulting image is illustrated in Figure 4.13. It can be observed that the influence of the IR images on the RGB images varies from image to image, depending on the surface temperature of the area.

Figure 4.13: The output of combining the RGB and IR images. It can be seen that the images taken at different location, weather conditions, and altitudes has large variety in their appearance.

4.3.3 Patches

The pre-processed images have a resolution of 3200×2400. Feeding the full-resolution images into the network would require significant computational resources, mainly GPU (graphical processing unit) requirements.

For rectangular images, YOLOv5 pads the image with zeros to make it square, so more GPU resources are needed for training. In making the square image, the original image gets padded with zeros in the smallest dimension to make it match the largest dimension.

One solution to this problem would be to reduce the resolution of the images. However, even though this would require fewer computational resources, this would also result in data loss. Since the problem is to find relatively small objects in the images, a loss of resolution was considered a suboptimal approach.

The approach was to divide the images and labels into smaller patches. Thus, the images were divided into twelve 800×800 patches. In this way, the resolution could be maintained. In addition, splitting the images allowed the network to be trained in larger batches on weaker GPUs. Square images bypass the problems of the images being padded to a square dimension during training, wasting GPU resources.

When the images were split, a label was assigned to a specific patch if the bounding box's center x and y coordinates were included in the image patch. In most cases, this results in a label that is more than 50% within the patch. In rare cases where the bounding box is in the corner of the image, the smallest possible area within the patch is more than 25%. Since the bounding

box is split between 4 images, the box with the center coordinates would always contain the most significant part.

The split was done after the dataset had been divided into training, validation, and test set, to ensure that the same images were in each dataset split.

4.3.4 Removing background images

Background images or blank images are images that do not contain objects for which the network is to be trained. These images help decrease the number of FPs detected by the network. It is considered optimal that the dataset contains 10% background images for helping to reduce the number of FPs detected by the network. When flying at a high altitude and using a camera field of view, large parts of the images often contain no sheep. As a result, the percentage of background images is much higher than recommended. Both have an impact on the training time of the network and also on the performance.

Since the images were captured from an altitude between 40 and 60 meters, large areas will not contain sheep in images with few subjects. Therefore, when splitting the dataset into patches, 10283 of the training images and 3569 of the validation images were background images without labels. Since 10% of background images is advantageous for training, background patches were randomly pruned until this number was achieved.

The background images were only pruned from the training and the validation set but left in the test set to ensure parity when testing the models using the different approaches to data pre-processing.

Figure 4.14: The figure shows the process of dividing the images into patches, and then removing the patches which contains no sheep. The removed patches are represented by the blue squares.

4.4 Label pre-processing

A previous master's thesis [12] showed that the network works well with white sheep but has significantly more problems with black, brown, and gray sheep. One possible explanation could be that this was due to the black and brown sheep being closer to the ground color and therefore harder to detect. However, this could not explain why gray sheep, naturally less well camouflaged and with a more similar appearance to white sheep, also had lower detection accuracy. An explanation could be that the number of black, brown, and gray sheep was too low in the dataset. To test this, two additional sets of labels were created:

One set of labels where there were two classes: 0 - colored sheep and 1 - white sheep. And a set of labels with only one class: 0 - sheep.

4.4.1 Overview over final datasets

The pre-processing of the images resulted in three different data sets, each with three different label pairs.

Dataset type	Train	Validation	Test	Total
Full	1143	404	135	1682
Pruned Patches	4119	1441	1620	7180

Table 4.3: The number of images in the different datasets.

The distribution of overall sheep images is as follows

	Black	Brown	Grey	White	Colored	All
Four classes	1806	913	2401	10433	5120	15553
Two classes	-	-	-	10433	5120	15553
One class	-	-	-	-	-	15553

Table 4.4: The distribution of the type of sheep in the total dataset. Labels colored sheep and all sheep are only used when training for two and one classes, respectively.

4.5 Machine Learning Model

For training the model, the PyTorch ML framework was chosen. PyTorch is an open-source ML framework developed by Facebook. Originally, PyTorch was a smaller competitor to Google AI's TensorFlow, but it quickly caught on and is now the leading framework. Hence, most modern models have a PyTorch implementation.

The network architecture chosen was the YOLOv5 model implemented in PyTorch was selected as [**glenn˙jocher˙2020˙4154370**]. This model was selected because it comes with state-of-the-art performance in object detection [5], open-source code, and ease of use and modification that comes with the implementation in PyTorch. In addition, the framework is expected to perform well on smaller datasets due to its data augmentation capabilities.

4.5.1 Data loader modification

YOLOv5 is commonly used for networks with three input channels but can scale its layers to handle four-channel input 'out of the box.' However, there are two problems with this approach. First, without further customization, YOLOv5 handles four-channel images by silently removing the last channel of the image. The second problem is that the 4-channel image formats the framework accepts are bulkier than the JPEG format initially used for the images. The conversion would lead to a dataset requiring more storage space than saving the RGB and IR images separately.

The data loader, responsible for reading and preprocessing the images, was modified to handle this issue. The modified version first loads the JPG image pair into memory. It then appends the single-channel IR image to the three-channel RGB image, creating a four-channel RGB-IR image where the infrared images are the fourth channel. This change allows the dataset to keep its smaller size while still utilizing the thermal image data.

4.5.2 Training

Access to the Idun cluster at IDI was granted for training the network, and all training was done on an NVIDIA Tesla V100 GPU with 32GB of VRAM [1]. The models were trained for the YOLOv5m network, which is a network with 21.2 million parameters. The network was chosen because of the lower training time than the YOLOv5l network. The large resolution images led to training only being possible for a batch size of 1 on the available hardware. Since one of the core features of YOLOv5 is the use of Mosaic augmentation [5], where multiple images from the same stack are merged, the potential benefits of using a larger model were outweighed by concerns about the impact on data augmentation. The YOLOv5m model was trained using a total of 18

different configurations. These configurations combine different input images and the numbers of different sheep classes. An overview of the configurations can be seen in table 4.5.

The models were trained for 100 epochs. The models were initialized with YOLOv5 models trained on the COCO dataset [58]. During training, the YOLOv5 uses data augmentation to improve generalization and avoid overfitting. The augmentations used and their weighting, as well as the other hyper-parameters used for the model, can be found in Appendix A.1.

Image Size	Full images									Paruned image patches								
Data Type	RGB			Combined			RGB-IR			RGB			Combined			RGB-IR		
Label Classes	4	2	1	4	2	1	4	2	1	4	2	1	4	2	1	4	2	1

Table 4.5: A overview of the different model configurations that were trained for this thesis

4.6 Source code

The source code used in this experiment can be found on GitHub. Image calibration [59]. Modified YOLOv5 [60]. Because of the large size of the dataset, it will be shared upon request.

Chapter 5

Results

After being trained for 100 epochs on an NVIDIA 32G TESLA GPU on the NTNU Idun cluster [1], the models were run on a test set of 135 images with ground truth for the full-size images. These were divided into $12 \times 135 = 1620$ images for the patches.

The metrics on which the model is evaluated are mAP, precision, and recall. Because of the way mAP is calculated, as seen in equation 3.6, comparing the different results between methods using different classes can be difficult. For example, since it is the mean, the effects of the classes are weighted equally regardless if four classes are used or two. The total mAP increases substantially if the lower performing classes are combined into a single class whose performance is close to its average. In this case, the highest-performing class would account for a much larger share of the average than before the other classes were combined. Thus, it is not sufficient to look at the difference in mAP to measure the performance of models trained in different classes. Since the problem in this work is finding something missing, the consequences of an FP are less than those of an FN. Therefore, special attention will be given to the number of TP detections and the models' recall value. As this thesis primarily aims to evaluate whether information can be gathered from the IR images, the inference time of the models is not considered.

5.1 Model Configurations

The models were trained with many different configurations in order to show the effect of different image sizes, input data types, and classes, as shown in table 4.5.

5.1.1 Image splitting and size

Half of the models were trained on full-size images 3200×2400 pixel, the other on image patches 800×800 pixels. Tables 5.1 and 5.4 show the mAP of the models trained and tested with full size images and image patches, respectively. It can be observed that the models trained with full-size images perform better compared to the patched models for both mAP_{50} and mAP_{95} in all configurations. The average difference is 0.77 for mAP_{50} and 0.07 for mAP_{95} in favor of the full-size images. The best performing model trained with patches scored 98.0 for mAP_{50} and 77.9 for mAP_{95}. Compared to the best performing model trained with the full-size images, this is a decrease of 0.9 in both categories.

The precision value 3.3 is a metric that indicates how many FPs the metric detects compared to the TPs. The precision values of the models for full-size images and for patches are given in the tables 5.2 and 5.2, respectively. 5.5, respectively. On average, the models trained with full-size images achieve 1.8 higher precision than those trained with image patches. For the best-performing models, the difference is 2.6. The recall value 3.4 measures how many FNs the model has. The value can be seen in table 5.3 for full-size images and in table 5.6 for image patches. The average

difference in the recall score is 0.7 in favor of the full-size images. For the best-performing models, this difference is 2.1 in favor of the full-size images.

On average, the models using patches perform worse on all metrics than those trained on full-size images. This decrease can likely be attributed to how the object labels were s when the patches were created. In some cases, it resulted in sheep in the image not being labeled if the center of the bounding box was not within the patch, leaving up to 50% of a sheep unlabeled. This fact impacts the mAP and precision scores but should be less present in the recall score. The labels alone would not lead to FNs but FPs.

The model trained on patches performs better on the combined image data than those trained on the full-size images. The reason for this is probably the color shift of the image. When the color of the black, brown, and gray sheep changes during the color shift, the distance between the classes decreases. So the sheep of the different classes will look more similar. For this reason, the network will confuse more sheep of one class with the others. Moreover, splitting the images into patches increases the spatial resolution of the image while maintaining the pixel resolution of a given area. Since this makes the spatial resolution of the sheep greater, the network can use different features at different stages [43]. If the data is noisy, this could give the models trained on patches a performance advantage over those trained on full-size images when the classification involves multiple classes. The same reasoning applies to why the patched images outperform the models on mAP_{95} when the data type was combined images and RGB IR images.

Because the models trained on full-size images perform better than those trained on patches, they will be emphasized in the following sections, where the impact of the different data types- and class configurations will be considered.

5.1.2 Image data type

The second configuration group is the data type. The models were trained using three different types of input images. The first type is RGB images which have three image channels. The second type is a combined RGB and IR image as shown in 4.13. These images also have three image channels. The final type of images is the RGB-IR, which are RGB and IR images concatenated by the data loader. These images have four image channels.

The models using only RGB images achieve higher mAP, precision, and recognition than those using combined images and the RGB-IR model. The model with the worst performance is the RGB-IR one.The average performance difference is 3.2 for mAP_{50} and 4.9 for mAP_{95}, in favor of the RGB models. For the models trained on combined data, the average difference is 2.7 and 3.7 in favor of the RGB models. For precision and recall, the average difference from the RGB model is 3.6 and 2.2 for the model using combined images. For the model using RGB-IR images, the difference is 3.4 and 4.7.

The decrease in performance when using IR images is surprising, considering Johansen's results on the combined use of RGB and IR images in sheep detection. These state that the model's accuracy increases when IR images are used. [12]. However, it should be noted that both the datasets and the network models used are different. The lower performance of the models trained on IR images compared to the RGB models suggests that the IR images may have too low a resolution to be used for sheep detection, as suggested by Bøckman [14]. The image quality is affected by the IR camera's low resolution and the sheep's surface temperature. The modest difference between sheep surface temperature, as reported by Guttormsen [15] is also observable in the dataset used for this thesis. The RGB-IR model differs the most from the RGB model and performs the worst on average. The comparatively higher performance of the combined image model can be explained by its input data being closer to the RGB images than that of the RGB-IR model. Therefore it is closer to the RGB model.

5.1.3 Number of classes

It can be seen from the results that combining the black, brown, and gray sheep into a single colored class leads to an mAP that is higher in all model configurations. The same holds for combining all classes into a single class. Although, as mentioned earlier, it is difficult to measure the effect of combining the classes using mAP, the fact that mAP is higher than for the individual classes suggests that there should be an overall increase in performance.

Full-size Images

Data type	Nr. classes	mAP (%)	Black	Brown	Grey	White	Colored	Total
RGB	4	50	94.5	86.1	91.8	98.5	-	92.7
		95	68.3	64.3	66.7	81.2	-	70.1
	2	50	-	-	-	97.6	94.6	96.1
		95	-	-	-	79.8	69.6	74.7
	1	50	-	-	-	-	-	**98.9**
		95	-	-	-	-	-	**78.8**
Combined	4	50	89.9	68.5	88.3	96.8	-	85.9
		95	61.0	48.8	60.3	77.0	-	61.8
	2	50	-	-	-	96.6	93.4	95.3
		95	-	-	-	78.9	65.2	73.9
	1	50	-	-	-	-	-	98.6
		95	-	-	-	-	-	76.7
RGB-IR	4	50	87.7	76.7	86.1	96.9	-	86.8
		95	63.8	55.8	60.9	78.4	-	64.7
	2	50	-	-	-	96.4	90.5	93.5
		95	-	-	-	76.3	62.9	69.6
	1	50	-	-	-	-	-	97.9
		95	-	-	-	-	-	74.6

Table 5.1: The results of training YOLOv5m for 100 epochs on a dataset with the full resolution images

Data type	Nr. classes	Black	Brown	Grey	White	Colored	Total
RGB	4	93.7	87.7	90.4	98.1	-	92.4
	2	-	-	-	97.1	94.5	95.8
	1	-	-	-	-	-	**98.7**
Combined	4	89.8	66.3	83.3	94.6	-	83.5
	2	-	-	-	97.4	91.4	94.4
	1	-	-	-	-	-	98.4
RGB-IR	4	88.7	69.8	88.5	95.0	-	85.5
	2	-	-	-	94.1	92.1	94.1
	1	-	-	-	-	-	97.3

Table 5.2: The precision of testing the YOLOv5m

Data type	Nr. classes	Black	Brown	Grey	White	Colored	Total
RGB	4	90.1	86.3	87.4	94.7	-	89.6
	2	-	-	-	94.5	92.2	93.3
	1	-	-	-	-	-	**96.8**
Combined	4	86.9	71.2	81.5	94.7	-	83.6
	2	-	-	-	93.9	92.4	93.2
	1	-	-	-	-	-	96.2
RGB-IR	4	83.9	81.2	77.7	93.7	-	84.1
	2	-	-	-	91.9	83.2	87.6
	1	-	-	-	-	-	93.9

Table 5.3: Recall score from running the YOLOv5m model trained on full images on the test set.

Image Patches

Data type	Nr. classes	mAP (%)	Black	Brown	Grey	White	Colored	Total
RGB	4	50	89.5	87.9	90.2	96.9	-	91.1
		95	62.9	64.2	64.8	78.9	-	67.7
	2	50	-	-	-	96.8	92.6	94.7
		95	-	-	-	79.4	68.7	74.1
	1	50	-	-	-	-	-	**98.0**
		95	-	-	-	-	-	**77.9**
Combined	4	50	86.5	76.8	86.4	96.4	-	86.5
		95	63.1	54.5	61.1	78.4	-	64.3
	2	50	-	-	-	97.0	92.6	94.8
		95	-	-	-	78.8	68.5	73.6
	1	50	-	-	-	-	-	97.7
		95	-	-	-	-	-	77.2
RGB-IR	4	50	86.3	80.4	86.2	96.2	-	87.3
		95	61.9	57.2	61.6	76.7	-	64.4
	2	50	-	-	-	95.7	89.8	92.8
		95	-	-	-	77.0	64.1	70.5
	1	50	-	-	-	-	-	96.3
		95	-	-	-	-	-	74.5

Table 5.4: The results of training YOLOv5m for 100 epochs on a dataset with images split into patches and background images pruned

Data type	Nr. classes	Black	Brown	Grey	White	Colored	Total
RGB	4	91.6	90.0	87.8	94.6	-	90.1
	2	-	-	-	93.8	89.4	91.6
	1	-	-	-	-	-	**96.1**
Combined	4	87.9	83.5	84.0	92.0	-	86.9
	2	-	-	-	93.8	90.7	92.2
	1	-	-	-	-	-	96.1
RGB-IR	4	84.0	83.4	84.3	92.4	-	86.0
	2	-	-	-	92.1	89.1	90.6
	1	-	-	-	-	-	94.1

Table 5.5: The Precision of testing the YOLOv5m for 100 epochs on the test spilt of the dataset with pruned patches

Data type	Nr. classes	Black	Brown	Grey	White	Colored	Total
RGB	4	87.1	90.0	85.7	93.9	-	89.2
	2	-	-	-	94.5	90.2	92.4
	1	-	-	-	-	-	94.4
Combined	4	84.1	76.2	83.6	94.8	-	84.7
	2	-	-	-	94.4	88.5	91.5
	1	-	-	-	-	-	**94.7**
RGB-IR	4	83.3	75.4	82.8	91.4	-	83.2
	2	-	-	-	92.7	85.1	88.9
	1	-	-	-	-	-	93.3

Table 5.6: The Recall from training the YOLOv5m for 100 epochs on the test split of the dataset with pruned patches

5.2 A closer look at model performance

YOLOv5 does not support subclasses. For this reason, the results of the models with merged classes do not provide information about the network's performance with different sheep colors. By comparing the detections of the model with the ground truth labels of the test set, the performance of such can be determined by classes. Since the ground truth labels also include the sheep color, the network's performance in the different classes can be evaluated. A prediction is considered TP if it has an IoU above 0.45. The predictions are evaluated without considering the predicted class. For example, if the network classifies a black sheep as a brown sheep, it is still considered a TP. The results of this prediction for the full models can be seen in Table 5.7.

Data type	Nr. classes	Black	Brown	Grey	White	Total	Found (%)	FP	FN
Testset	4	131	80	238	915	1364	-	-	-
RGB	4	129	78	237	902	1346	98.6	31	18
	2	127	77	236	895	1335	97.9	42	29
	1	128	76	237	896	1337	98.0	40	27
Combined	4	122	76	234	893	1325	97.1	51	39
	2	125	74	235	900	1334	97.8	38	30
	1	124	74	233	891	1322	96.9	45	42
RGB-IR	4	117	74	229	890	1310	96.0	37	54
	2	112	70	225	888	1295	94.9	71	69
	1	113	72	219	881	1305	95.7	59	79

Table 5.7: The table shows the number of sheep found by each model in the detection step. The test row show how many sheep are labeled in the test set. FP and FN are false positives and false negatives respectively.

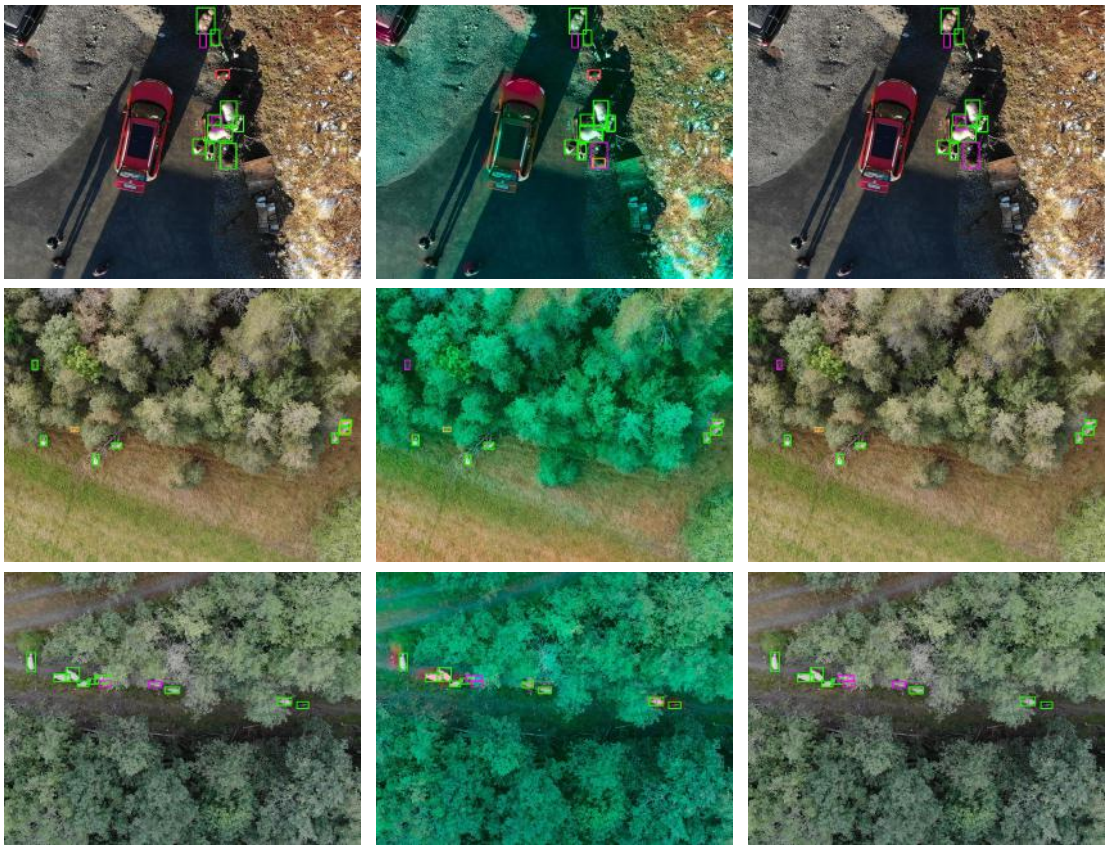
As shown from the table 5.7, the models with the largest mAP, highest precision, and highest recall do not exclusively outperform the other models, as is the case when looking at the mAP. However, the same trend in terms of data type is also evident in the results, where the corresponding RGB model outperforms the combined models, which in turn outperforms the RGB-IR model.

Particular attention should be paid to the combined model for all classes. It has the worst mAP score of all the trained models but is much closer to the other models in evaluating successful detections. When the color space shifts, the difficulty in distinguishing between black, brown, and gray sheep may lead to misclassification but not false detection.

Even more surprising is the finding that the model trained in all classes is the most successful model for localizing sheep. It applies to both the RGB and RGB-IR models. This model also has the lowest number of FNs on the test set, despite using the class configuration that performs worst on mAP, precision, and recall.

The best performing model is the RGB model trained on all classes. This model achieves a detection rate of 98.6 % and detects only 2.3 % of FPs. Comparably the RGB model trained on a single class which reached the highest score in total mAP, precision, and recall, has a detection rate of 98 % and detects 2.9 % FPs. The best performing model using combined images was the one trained on two classes of sheep. This model achieves a detection rate of 97.8 % and detects 2.8 % of FPs. Similar to the RGB model, the model trained with all four classes of sheep performed best with the RGB-IR images. This model has a detection rate of 96.0 % and detects false 2.7 % FPs.

In figure 5.1 a comparison between detections from the best performing models for RGB, combined, and RGB-IR images is shown. It can be seen that even the RGB model is more accurate also in cases where the sheep are occluded or hard to see because of the light conditions. The reason for these differences is that metrics are calculated without considering the confidence score. However, by looking at the results in 5.7 and the detection in 5.1, this seems to have little impact on the success rate of finding the sheep.



(a) RGB (b) Combined (c) IR

Figure 5.1: A comparison between the best performing RGB, Combined, and RGB-IR image models. Correct detections are marked with a green box, missing detections with a purple box, and boxes of other colors signify a false detection.

5.3 Analysis of the results RGB-IR images

Since the performance of the IR image model is the lowest of the three model types across all classes, a closer look is taken at how the fourth IR channel affects the model. One possible explanation would be that the model learns to ignore the data from the IR images and give more weight to the other three channels. For testing this, the IR images are replaced with black images of the same dimension. The result is a black image, called a blank image. Table 5.8 shows the results of replacing the IR images with blank images and then running the models on the test set.

It can be seen that the IR images significantly affect the model's performance. Especially for the minority classes, the performance difference is more significant than for white sheep. The most considerable difference of -18.2 is detecting colored sheep as a single class with IR images and blank IR images. This could be because the white sheep's IR images are less critical for detection due to the larger amount of training data, as shown in Table 4.2. However, this is not confirmed by the results in Table 5.8, where the combined class of colored sheep shows a larger drop in performance than any of the three classes. Surprisingly, the white sheep class has a significantly smaller performance drop for the model trained together with the colored sheep of the three different classes than when trained together with the solid colored sheep class, with a difference of 2.2% for mAP_{50} and 3.1% for mAP_{95}. The models appear to depend more on the IR images when the two-class configuration is used than the one-class configuration. This contradicts the theory that the model discards the IR images when it accesses more training data. One possible explanation is that the IR data for sheep of different classes are very similar. Therefore, when training as separate classes, the network receives negative reinforcement for data similar to data that would result in positive reinforcement in other cases. For this reason, the network responds

Data type	Nr. classes	mAP (%)	Black	Brown	Grey	White	Colored	Total
Blank IR	4	50	82.7	68.4	77.6	95.2	-	81.0
		95	58.5	48.4	52.3	73.7	-	58.2
	2	50	-	-	-	92.5	72.3	82.4
		95	-	-	-	68.5	47.0	57.7
	1	50	-	-	-	-	-	92.6
		95	-	-	-	-	-	64.6
Difference	4	50	-5.0	-8.3	-8.5	-1.7	-	-5.8
		95	-5.3	-7.4	-8.1	-4.7	-	-6.5
	2	50	-	-	-	-3.9	-18.2	-11.1
		95	-	-	-	-7.8	-15.9	-11.9
	1	50	-	-	-	-	-	-5.3
		95	-	-	-	-	-	-10.0

Table 5.8: The results of testing the YOLOv5m models on a dataset with the full resolution images with IR images replaced with blank black images contain only zeros.

more weakly to the IR images.

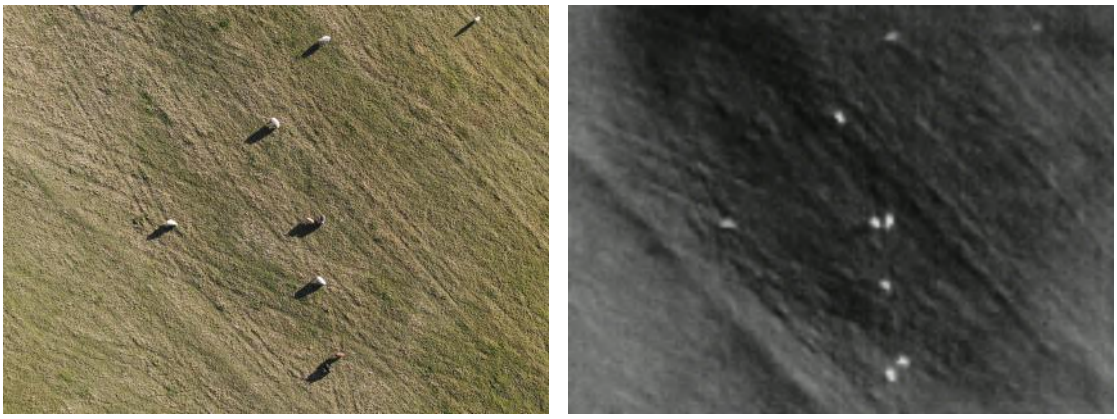


Figure 5.2: A comparison between the IR image of white and black sheep on a sunny day. While it can be seen that the black sheep is somewhat more visible within the image compared to the white sheep, the difference is rather small. This image is close to the ideal situation where this would make a difference. The image is taken in an open field on a cold day where the sun reflects off the sheep.

5.4 Sources of error

5.4.1 Data collection

The drone, DJI Mavic Enterprise Dual, used for data collection has a setting that limits the temperature range captured by the IR image sensor. The pixels in the resulting IR image are normalized to this range, meaning that when a smaller range is set, slight temperature differences within the range are more visible in the image. For a small number of images, this setting appears to have been tweaked by the different groups operating the drone. This leads to inconsistency in the IR images of the dataset, which is likely to affect the model's training negatively.

5.4.2 Dataset bias

The results show that all networks perform best with white sheep and have larger difficulty with colored sheep, predominantly brown sheep. Presumably, this is due to the fact that sheep are more similar in color to the terrain and therefore more difficult to detect and that sheep are largely underrepresented in the data set.

5.4.3 Image quality and dataset curation

Some images had a large discrepancy between the position of the sheep in the IR and RGB images. The reason for this is a slight delay in capturing images with the two image sensors. Therefore, whenever the sheep move or the camera drone turns or moves while capturing the photo, a discrepancy exists between the position of the sheep in the image pairs.

Other images in the dataset are of such low quality that it is difficult to extract useful information. These are images taken while the drone was moving or because lighting conditions were poor. Such images should be removed before training the network. Even though the model might encounter such images in a real-world context, they are likely to be more challenging to train. In addition, they negatively affect most images taken under better conditions.

5.4.4 Image labeling

The quality of the labels for the dataset also has to be questioned. Four different class labels, black, brown, grey, and white, were used when the dataset was labeled. It is sometimes hard to know in which category to label a sheep. The sheep are often multicolored; even when they are unicolored, it can be hard to decide when the darkest brown and grey sheep should be labeled as black. With this comes much human error when the labeler has to make a call using their intuition. In the process of labeling this dataset, no clear guidelines were set other than to follow the trend of the preexisting dataset. While intuitive, it is unknown if this is the ideal way to classify the sheep.

Missing labels and overconfident labels

When the sheep are located in busy terrain or partly occluded, it can be hard to see them for the labelers. This will lead to some sheep being missed and thus not labeled. It was observed in the test set that the models detected sheep that were not labeled. Because of this, they were counted as FPs in the model evaluation.

It is also a challenge to decide how much of a sheep needs to be visible to be labeled. The labelers have prior knowledge that a sheep is highly likely to be present in the image. If the images are part of a series, the position of a sheep in the previous image might help the labeler find the sheep in the next image. While it is attempted to consider each image independently, the decision depends on intuition, which the abovementioned factors can subconsciously influence. For the model, this can create "unfair" working conditions where it is not reasonable or desirable to learn from the labeled object.

False labels

In an attempt to train the model to avoid FP detections, many images in the dataset deliberately contain objects that will look like sheep at first glance. There are occurrences of sheep-looking rocks, tree stumps, and lampposts in the dataset. When combined with the previous problem of sheep being hard to locate, the labeler might also sometimes get fooled by these items. Occurrences of incorrectly labeled objects were present in the test set. It is unknown to which extent this occurs in the whole dataset.

5.4.5 Are the IR models detecting hidden sheep?

Another possible source of error is that sheep located in the forest might suffer from the missing label problem. It is difficult for humans to detect the sheep when the trees largely occlude the sheep. As a result, sheep hidden in the brush could be detected by the model trained on the IR images but counted as FPs because they were not labeled in the dataset.

When inspecting the FPs output by the network, this appears unlikely. The observable common denominator for the sheep that are not detected in the RGB-IR and RGB images is that sheep are somehow occluded. As shown in figure 5.1, in selected scenarios where the sheep are partly covered by vegetation and foliage, the pure RGB network detects the sheep better than the IR network. It is more likely that the main reason for the difference is that the IR images are of low resolution and are just as, if not more, affected by occlusion than the RBG images.

5.4.6 Dataset shift and random dataset splitting

The dataset consists of many images taken in quick succession. The result is many similar images. This was not considered when splitting the dataset into training, validation, and test sets. As a result, the model may have been trained on something similar to the test dataset, resulting in a model that might be very specifically tuned to this group of sheep, terrains, and light conditions.

5.5 Weighing the pros and cons of using a drone with IR

As was seen in the result section, the models trained on RGB images only consistently outperformed the models trained on RGB and IR images. It should be noted that the model's effectiveness in detecting sheep in RGB images is very high. Since the network can learn so well from the RGB images, the IR images may be redundant in cases where they provide good data independently. If there had been more cases where the sheep were difficult to locate in low light conditions, the IR images might have had more of a positive impact. However, it is questionable if this is a good use case for the farmer. If the drone is manually operated, the operator should have a direct line of sight to the drone. It is also questionable whether farmers would want to search for sheep at times when light conditions are not ideal for responding to the drone's detection. Sheep gathering is often done on foot in rough terrain. This work can be done most efficiently and safely in daylight. If IR images offer an advantage over RGB-only imagery when flying at night, that does not necessarily mean that the IR data is particularly good. For these images to be useful, they must provide unobtainable data during daylight and are of such high quality that it would lead to detections of sheep that would otherwise not be found. Looking at the data in the tables 5.1 and 5.4, this seems like an unlikely scenario.

Chapter 6

Conclusion and Future Work

The research showed that IR images negatively influenced the detection of sheep in the images. The cause was both cases where the sheep were partially obscured by vegetation or in open fields. However, the accuracy of the predictions is uncertain because although there were no duplicate images in the training dataset and the test, there will still have been cases with very similar images due to the nature of the dataset having a small number of different sheep.

Does training on IR images improve model performance overall?

The impact of using IR data for training and detection with the model is negative. The models trained using IR images performed worse on all metrics considered in this thesis. The IR camera works best to locate sheep in stable and cold weather conditions. In these cases, the sheep are more visible in the IR images. However, the IR data is not stable enough to give a good result when using the camera in different weather conditions and temperatures.

RQ2: Does training on IR images help find sheep occluded by brush and vegetation in forested areas?

Using IR imagery did not have a positive effect when detecting obscured sheep. It was observed that these sheep were where the models using IR images performed the worst. This drop in performance is because a low-resolution IR camera is not more resistant to occlusion than a high-resolution RGB camera. Therefore, it can be concluded that the IR imagery of the standard used in this work does not positively affect the detection of sheep obscured by light to dense foliage or vegetation.

RQ3: Is the quality of the IR data good enough to justify the extra cost of an IR image sensor?

Based on the results, an IR image sensor of the standard used in this thesis does not provide an advantage over high-resolution RGB image sensors. Furthermore, the DJI Mavic 2 Enterprise is an expensive piece of equipment that may present an entry barrier for a framer who wants to use this technology to locate sheep. Alternative camera drones without an IR camera offer a similar or higher camera resolution at a lower price point. Considering the comparison of drones in 2.1, the DJI Mavic Air 2 offers a higher resolution RGB camera and improved flight time at about $\frac{1}{4}$ of the cost. Therefore, when considering the findings of this thesis, the option of using a lower-cost drone without an IR camera sensor is recommended. The door should, however, be held open for using a higher resolution IR image sensor in the future.

6.1 Future Work

In the Results section, it can be seen that using a 4-channel network did not increase the model's performance. Previous approaches have shown that using fusion networks is successful [12]. Incorporating this method into the YoloV5 model could lead to better results.

This project also did not explore using reasonably similar data sets and transfer learning. Part of the reason is that no datasets with infrared and color image pairs were found. However, finding such a dataset could be explored further. In this case, pre-training with such a dataset and then fine-tuning the network for the sheep images could be considered.

All images in the current dataset were acquired from a 90-degree angle. While this is convenient for seeing through light forests and other vegetation, it also leads to situations where the sheep are obscured. However, this would not be the case with a different camera angle. An example is a sheep standing on the edge of a dense spruce forest. It is impossible to see the sheep directly from above with both the thermal imaging camera and the color camera in these cases. Additionally, images taken from a 90-degree angle severely limit the drone's field of view and require flight time to cover an area. Therefore, it might be beneficial to expand the dataset to include images of sheep taken from different angles and distances and attempt to train the model to generalize between networks. That potentially could lead to a more versatile and accurate detector covering larger areas.

If steps are taken to expand the dataset with images from different angles, an effort will be needed to determine the distance boundary for sheep detection. It would also be beneficial to investigate how different distances affect detection accuracy.

In addition, further data extension could be experimented with to find occluded and different colored sheep. As can be seen from the results, black, brown, and multicolored sheep are more difficult to detect than white. The reason could be that colored sheep are underrepresented in the data set, and their darker colors are more similar to the ground. Consequently, it would be beneficial to find a way to expand the dataset to include more data that resemble the underrepresented sheep. One solution could be to expand within the bounding box by taking the negative of the color values above a certain threshold so that the pixels of a white sheep would ideally be a darker color.

Distinguishing between hidden and visible sheep was planned but discarded due to the small number of these types of sheep and the difficulty of accurately labeling them. The procedure would require labels to locate nearly invisible sheep in the color images, but these could be determined with certainty to be sheep when juxtaposed with the infrared images. Unfortunately, the poor quality of IR images makes this challenging unless it is known where the sheep are located in the images. With a higher quality IR camera, this could be a valuable area to explore in the future.

Bibliography

1. Själander, M., Jahre, M., Tufte, G. & Reissmann, N. EPIC: An Energy-Efficient, High-Performance GPGPU Computing Research Infrastructure. *arXiv:1912.05848 [cs].* arXiv: 1912.05848 [cs] (May 2019).
2. Johnson, S. P. Visual development in human infants: Binding features, surfaces, and objects. *Visual Cognition* **8,** 565–578. eprint: https://doi.org/10.1080/13506280143000124. https://doi.org/10.1080/13506280143000124 (2001).
3. Szeliski, R. *Computer Vision - Algorithms and Applications, Second Edition* Available at https://szeliski.org/Book/ (Springer, 2022).
4. Voulodimos, A., Doulamis, N., Doulamis, A. & Protopapadakis, E. Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience* **2018** (2018).
5. Nepal, U. & Eslamiat, H. Comparing YOLOv3, YOLOv4 and YOLOv5 for Autonomous Landing Spot Detection in Faulty UAVs. eng. *Sensors (Basel, Switzerland)* **22,** 464. ISSN: 1424-8220 (2022).
6. Gonzalez, L. F. *et al.* Unmanned aerial vehicles (UAVs) and artificial intelligence revolutionizing wildlife monitoring and conservation. eng. *Sensors (Basel, Switzerland)* **16,** 97. ISSN: 1424-8220 (2016).
7. Rey, N., Volpi, M., Joost, S. & Tuia, D. Detecting animals in African Savanna with UAVs and the crowds. eng. *Remote sensing of environment* **200,** 341–351. ISSN: 0034-4257 (2017).
8. Sarwar, F., Griffin, A., Rehman, S. U. & Pasang, T. Detecting sheep in UAV images. eng. *Computers and electronics in agriculture* **187,** 106219. ISSN: 0168-1699 (2021).
9. Dian Bah, M., Hafiane, A. & Canals, R. Deep learning with unsupervised data labeling for weed detection in line crops in UAV images. eng. *Remote sensing (Basel, Switzerland)* **10,** 1690. ISSN: 2072-4292 (2018).
10. Anna Blix, O. V. (*Sauehold i norge* https://snl.no/sau (2nd June 2022).
11. Muribø, J. H. *Locating Sheep with YOLOv3* eng. 2019. http://hdl.handle.net/11250/2619041.
12. Johansen, K. M. *Towards Improved Sheep Roundup - Using Deep Learning-Based Detection on MultiChannel RGB and Infrared UAV Imagery* 2020. https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2779322.
13. Furseth, O. K. & Granås, A. O. *Real-time Sheep Detection - Improving Retrieval of Free-ranging Sheep Using Deep Learning-based Detection on Drone Imagery Running on Mobile Devices* eng. 2021. https://hdl.handle.net/11250/2834579.
14. Sørensen Bøckman, H. *Locating sheep in the highlands with aerial footage and a lightweight algorithm system.* eng. 2021. https://hdl.handle.net/11250/2984882.
15. Guttormsen, M. *Gjenfinning av sau ved hjelp av drone* nor. 2019. https://hdl.handle.net/11250/2656671.
16. Lee, S., Song, Y. & Kil, S.-H. Feasibility analyses of real-time detection of wildlife using uav-derived thermal and rgb images. eng. *Remote sensing (Basel, Switzerland)* **13,** 2169. ISSN: 2072-4292 (2021).
17. Kolltveit, G. Animal bells in early scandinavian soundscapes. *Studies in music archaeology VI. Current challenges and new objectives in music archaeology,* 147–153 (2008).

18. Knarrum, V. *et al.* Brown bear predation on domestic sheep in central Norway. *Ursus* **17,** 67–74 (2006).

19. Caja, G. *et al. State of the art on electronic identification of sheep and goat using passive transponders* in *Data collection and definition of objectives in sheep and goat breeding programmes: new prospects. Proc. of the meeting of the FAO-CIHEAM Network of Cooperative Research on Sheep and Goats, Subnetwork on Animal Resources, jointly organized with INRA-SAGA, Toulouse (France)* (1997), 43–57.

20. Edwards, D., Johnston, A. & Pfeiffer, D. A comparison of commonly used ear tags on the ear damage of sheep. *Animal Welfare* **10,** 141–151 (2001).

21. Nofence. *What is nofence?* https://www.nofence.no/en/what-is-nofence (2nd June 2022).

22. Telespor. *Telespor - Produkt* https://telespor.no/produkt/ (2nd June 2022).

23. Findmy. *Findmy - Produkt* https://www.findmy.no/nb/funksjoner (2nd June 2022).

24. Smartbjella. *Smartbjella Sporing* May 2021. https://smartbjella.no/produkt/.

25. Wester-Ebbinghaus, W. Aerial photography by radio controlled model helicopter. *The Photogrammetric Record* **10,** 85–92 (1980).

26. Quan, Q. *Introduction to multicopter design and control* (Springer, 2017).

27. DJI. *DJI Mavic Enterprice 2 Dual Specifications* https://www.dji.com/no/mavic-2-enterprise/specs (30th Mar. 2022).

28. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521,** 436–444. ISSN: 1476-4687. https://doi.org/10.1038/nature14539 (May 2015).

29. Mazur-Milecka, M. & Ruminski, J. Deep learning based thermal image segmentation for laboratory animals tracking. eng. *Quantitative infrared thermography* **18,** 159–176. ISSN: 1768-6733 (2021).

30. Ghosh, S., Das, N., Das, I. & Maulik, U. Understanding deep learning techniques for image segmentation. *ACM Computing Surveys (CSUR)* **52,** 1–35 (2019).

31. Bodapati, J. D. & Veeranjaneyulu, N. Feature extraction and classification using deep convolutional neural networks. *Journal of Cyber Security and Mobility,* 261–276 (2019).

32. LeCun, Y. *et al.* Learning algorithms for classification: A comparison on handwritten digit recognition. *Neural networks: the statistical mechanics perspective* **261,** 2 (1995).

33. Nielsen, M. A. *Neural Networks and Deep Learning* misc. 2018. http://neuralnetworksanddeeplearning.com/.

34. LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE* **86,** 2278–2324 (May 1998).

35. Girshick, R., Donahue, J., Darrell, T. & Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. eng (2013).

36. Ren, S., He, K., Girshick, R. & Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. eng. *IEEE transactions on pattern analysis and machine intelligence* **39,** 1137–1149. ISSN: 0162-8828 (2017).

37. He, K., Gkioxari, G., Dollar, P. & Girshick, R. Mask R-CNN. eng. *IEEE transactions on pattern analysis and machine intelligence* **42,** 386–397. ISSN: 0162-8828 (2020).

38. Redmon, J., Divvala, S., Girshick, R. & Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. eng (2015).

39. Girshick, R. *Fast R-CNN* eng. in *2015 IEEE International Conference on Computer Vision (ICCV)* **2015** (IEEE, 2015), 1440–1448. ISBN: 1467383910.

40. Redmon, J. & Farhadi, A. YOLO9000: Better, Faster, Stronger. eng (2016).

41. Redmon, J. & Farhadi, A. YOLOv3: An Incremental Improvement. eng (2018).

42. Redmon, J. *I stopped doing CV research (...)* https://twitter.com/pjreddie/status/1230524770350817280?lang=en (27th May 2022).

43. Bochkovskiy, A., Wang, C.-Y. & Liao, H.-Y. M. YOLOv4: Optimal Speed and Accuracy of Object Detection. eng (2020).

44. Wang, C.-Y. *et al.* CSPNet: A New Backbone that can Enhance Learning Capability of CNN. eng (2019).

45. Huang, G., Liu, Z. & Weinberger, K. Q. Densely Connected Convolutional Networks. *CoRR* **abs/1608.06993.** arXiv: 1608.06993. http://arxiv.org/abs/1608.06993 (2016).

46. Zheng, Z. *et al.* Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression. eng (2019).

47. Jocher, G. *et al. ultralytics/yolov5: v3.1 - Bug Fixes and Performance Improvements* version v3.1. Oct. 2020. https://doi.org/10.5281/zenodo.4154370.

48. *YOLOV5* 2022. https://blog.csdn.net/Q1u1NG/article/details/107511465 (6th June 2022).

49. Salman, S. & Liu, X. Overfitting mechanism and avoidance in deep neural networks. *arXiv preprint arXiv:1901.06566* (2019).

50. Gavrilov, A. D., Jordache, A., Vasdani, M. & Deng, J. Preventing model overfitting and underfitting in convolutional neural networks. *International Journal of Software Science and Computational Intelligence (IJSSCI)* **10,** 19–28 (2018).

51. Torralba, A. & Efros, A. A. *Unbiased look at dataset bias* eng. in *CVPR 2011* (IEEE, 2011), 1521–1528. ISBN: 1457703947.

52. NVIDIA. *Nvidia V100* https://www.nvidia.com/en-us/data-center/v100/.

53. DJI. *DJI Mavic 2 Enterprise (Dual)* 2022. https://djioslo.no/produkt/mavic-2-enterprise/dji-mavic-2-enterprise-dual/#lg=1&slide=0 (6th June 2022).

54. Animalia. *Saueraser og Ulltyper* Sept. 2021. https://www.animalia.no/no/Dyr/ull-og-ullklassifisering/saueraser-og-ulltyper/ (7th June 2022).

55. Roboflow. *Give your software the power to see objects in images and video* https://roboflow.com/.

56. Sadekar, K. *VirtualCam* https://github.com/kaustubh-sadekar/VirtualCam. 2020.

57. Van der Walt, S. *et al.* scikit-image: image processing in Python. *PeerJ* **2,** e453 (2014).

58. Lin, T.-Y. *et al. Microsoft COCO: Common Objects in Context* 2014. https://arxiv.org/abs/1405.0312.

59. Stemshaug, H. *ImageCalibration* https://github.com/Hallvardd/ImageCalibration. 2022.

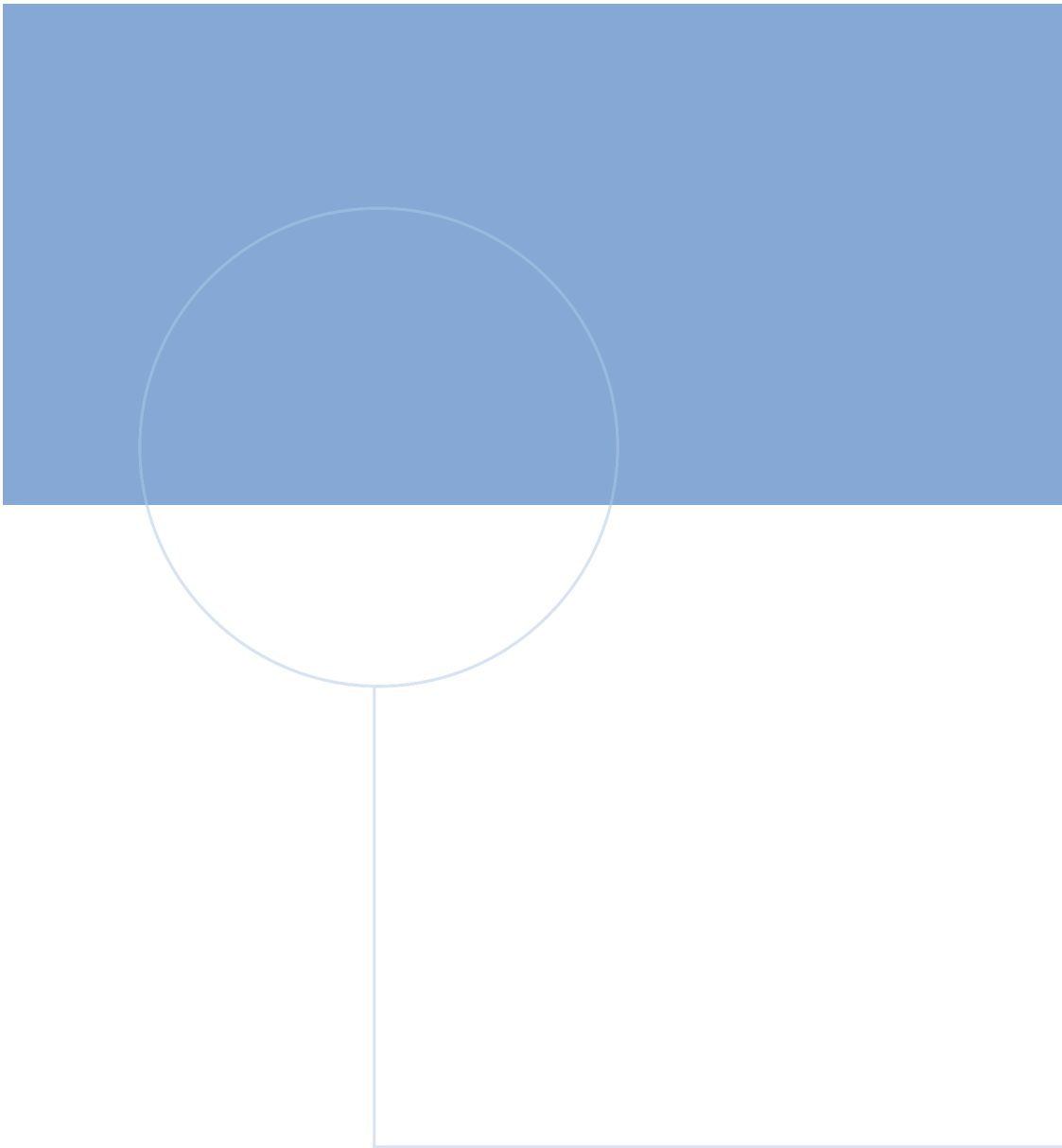
60. Stemshaug, H. *Woolov5* https://github.com/Hallvardd/woolov5. 2022.

Appendix A

Appendix

A.1 YOLOv5 Hyperparameters

```
# Hyperparameters for COCO training from scratch
lr0: 0.01  # initial learning rate (SGD=1E-2, Adam=1E-3)
lrf: 0.2  # final OneCycleLR learning rate (lr0 * lrf)
momentum: 0.937  # SGD momentum/Adam beta1
weight_decay: 0.0005  # optimizer weight decay 5e-4
warmup_epochs: 3.0  # warmup epochs (fractions ok)
warmup_momentum: 0.8  # warmup initial momentum
warmup_bias_lr: 0.1  # warmup initial bias lr
box: 0.05  # box loss gain
cls: 0.5  # cls loss gain
cls_pw: 1.0  # cls BCELoss positive_weight
obj: 1.0  # obj loss gain (scale with pixels)
obj_pw: 1.0  # obj BCELoss positive_weight
iou_t: 0.20  # IoU training threshold
anchor_t: 4.0  # anchor-multiple threshold
# anchors: 0  # anchors per output grid (0 to ignore)
fl_gamma: 0.0  # focal loss gamma (efficientDet default gamma=1.5)
hsv_h: 0.015  # image HSV-Hue augmentation (fraction)
hsv_s: 0.7  # image HSV-Saturation augmentation (fraction)
hsv_v: 0.4  # image HSV-Value augmentation (fraction)
degrees: 0.0  # image rotation (+/- deg)
translate: 0.1  # image translation (+/- fraction)
scale: 0.5  # image scale (+/- gain)
shear: 0.0  # image shear (+/- deg)
perspective: 0.0  # image perspective (+/- fraction), range 0-0.001
flipud: 0.0  # image flip up-down (probability)
fliplr: 0.5  # image flip left-right (probability)
mosaic: 1.0  # image mosaic (probability)
mixup: 0.0  # image mixup (probability)
```



NTNU

Norwegian University of
Science and Technology