Ingebrigt Nygård and Sebastian Vittersø

Improved Sheep Detection

Modifying YOLOv5 to accurately detect grazing sheep in UAV imagery

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

Master's thesis

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

NTNU
Kunnskap for ei betre verd

Ingebrigt Nygård and Sebastian Vittersø

Improved Sheep Detection

Modifying YOLOv5 to accurately detect grazing sheep in UAV imagery

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

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

Abstract

Sheep farmers in Norway are in sore need of modernization. The farmers often use rangelands as grazing areas for their sheep, and during grazing season the farmer must conduct weekly inspections of their herd, which currently is a manual task. At season's end, the sheep must be located and collected, and the farmer often needs assistance from their family, friends and neighbors for this. The weekly inspections and the localizing of sheep are time consuming tasks, but an alleviation for the farmer might come with the utilization of autonomous drones, combined with the automatic image processing capabilities of modern machine learning.

This master thesis focuses on the machine learning aspects of a system solving this challenge. It will consider what changes can be made to the state-of-the-art object detection networks to further improve the network's results. All testing is done using variations of the network architecture YOLOv5, applied to a dataset of relevant RGB and IR imagery of sheep, captured by a remote controlled UAV. Through rigorous examination, the thesis tests what effects three types of model variations will have on the results. Firstly, the model size, an editable parameter in the YOLOv5 framework, is varied to see to what degree the lowered computing demands of a smaller model will degrade the accuracy of the predictions made. Secondly, changes are made to the model architecture to include IR data. Two new variants are proposed: One variant applies a fusion based architecture which processes the RGB and the IR image in separate pipelines before fusing the intermediary data after the backbone. Another variant only varies slightly from the default RGB-only architecture, by accepting a 4 channel input in the format RGBI, and processing just as it would before. Thirdly, the final variation relates to the preprocessing of the image data and the postprocessing of the results produced by the network. It is examined whether a model accepting multiple smaller image tiles performs better than one without this tiling preprocessing, and a system is designed to combine the tiled predictions into full image predictions, to compare the results. All possible combinations of these three variations are trained and tested on two separate datasets. There are clear trends towards improved accuracies when using IR imagery, and similar, though less clear trends can be seen when using tiled images. It's worth noting that the latter also drastically increases processing times. The smaller models almost performed as well as the large ones, but there is a slight increase in accuracy when the larger models are used. As all models are able to perform the necessary processing within the given time frames, given proper hardware, these models are all usable in practical applications.

The implementation of a system using these principles is not only viable, but might be a necessary path towards more sustainable husbandry and agricultural practices in the future.

Sammendrag

Sauebønder i Norge har et sårt behov for modernisering. Bøndene bruker ofte utmark som beiteområder for sauene sine, og i beitesesongen er bonden lovpålagt å gjennomføre ukentlige inspeksjoner av besetningen, noe som i dag er en manuell oppgave. Ved sesongslutt skal sauene lokaliseres og samles, og bonden trenger ofte bistand fra familie, venner og naboer til dette. De ukentlige inspeksjonene og lokaliseringen av sau er tidkrevende, men en hjelp for bonden kan være å ta i bruk autonome droner, kombinert med de automatiske bildebehandlingsmulighetene i moderne maskinlæring.

Denne masteroppgaven fokuserer på maskinlæringsaspektene ved et system som løser denne utfordringen. Den vil vurdere hvilke endringer som kan gjøres i objektdeteksjonsnettverkene for å forbedre nettverkets resultater ytterligere. All testing gjøres med variasjoner av nettverksarkitekturen YOLOv5, brukt på et datasett med relevante RGB- og IR-bilder av sauer, tatt av en fjernstyrt drone. Gjennom grundig undersøkelse tester oppgaven effekten tre forskjellige variasjoner har på resultatene. For det første har modellstørrelsen, en redigerbar parameter i YOLOv5-rammeverket, blitt variert for å se i hvilken grad de reduserte datakravene til en mindre modell vil forringe nøyaktigheten til prediksjonene som er gjort. For det andre gjøres det endringer i modellarkitekturen for å kunne inkludere IR data. To nye varianter foreslås: En bruker en fusjonsbasert arkitektur som behandler RGB- og IR-bildet i separate *pipelines* før den fusjonerer dataene etter *backbonen*. En annen varierer bare litt fra den vanlige RGB-arkitekturen ved å akseptere 4-kanals *input* på et RGBI-format, og deretter behandle det akkurat som før. For det tredje er den siste variasjonen knyttet til preprosessering av bildedataene og postprosessering av resultatene produsert av nettverket. Det undersøkes om en modell som aksepterer flere mindre, flislagte bilder yter bedre enn en modell uten denne preprosesseringen. En algoritme er utviklet for å sammenligne resultatene ved å kombinere de mindre prediksjonene til fullbildeprediksjoner. Alle de mulige kombinasjonene av disse tre variantene er trent og testet på to separate datasett. Det er klare trender mot forbedret nøyaktighet ved bruk av IR-bilder, også noe forbedring ved bruk av flislagte bilder, selv om sistnevnte også drastisk øker prosesseringstiden. De mindre modellene presterte nesten like bra som de store, men det vises en liten økning i nøyaktigheten når de større modellene brukes. Ettersom alle modellene er i stand til å utføre nødvendig prosessering innenfor gitte tidsrammer, gitt riktig maskinvare, er alle de testede modellene brukbare i praktiske applikasjoner.

Implementasjonen av et system som bruker disse prinsippene er ikke bare realistiske, men kan være en nødvendig vei mot et mer bærekraftig husdyrhold og landbrukspraksis i fremtiden.

Preface

This Master's thesis contains the results of research conducted in spring 2022, and it is written for the Department of Computer Science (IDI) at the Norwegian University of Science and Technology (NTNU) in Trondheim.

This thesis is part of a series of theses written for IDI, focusing on the improvement and modernization of sheep farming. The thesis focuses its attention on the application of modern neural networks for object detection within the area of sheep retrieval and monitoring, and the effects of modifications to these neural networks. The work is based on initial work done in fall 2021, during our specialization project [1]. As we do not expect the reader to have read through that, and as it is written in Norwegian, some of the theory and key points in some method sections have been included in this thesis as well.

We would like to thank Svein-Olaf Hvasshovd, for supervising our work through weekly meetings. Subsequently, we would like to thank our loving and supportive wives, and to offer our appreciation to Christian Frøystad for reviewing our work and assisting us in the writing process. A thank-you is due to Hallvard Stemshaug for his work in extending the data set on which this thesis rests; the data has been invaluable.

Contents

Figures

Tables

Chapter 1

Introduction

In this first chapter the background and motivation for the thesis are presented. The chapter also puts forward the overall goal for the thesis, and research questions the thesis aims to answer. Finally, the scopes and limitations of the thesis are presented.

1.1 Background and Motivation

There is a long tradition in Norway of releasing sheep to graze freely in rangelands throughout the summer season. This tradition allows the sheep owners to take advantage of the otherwise unused food resources [2]. The sheep farmers have on average 40% of their food resources from the rangelands [3]. According to Landbruksdirektoratet, approximately 11,000 distinct agricultural holdings released 1.9 million sheep into the rangelands in 2021 [4]. Norwegian authorities require farmers to perform weekly inspections throughout the grazing period and keep count of the sheep scattered in the rangelands [5]. These inspections are often done as a collaborative effort by multiple sheep farmers, where they take turns overseeing the rangelands. This process is quite time consuming, and it is difficult to get a complete overview of the entire herd.

Another challenge is to find the sheep scattered in the grazing area at summer's end. This has traditionally been a manual task, covering the grazing area by foot looking for sheep and bringing them back to the farm. Thousands of farmers, assisted by their families and friends, come together to round up the sheep in September and October. In spite of this collaboration, the farmers generally do not find all the sheep that were sent to graze in the rangelands. According to the statistics reported to Landbruksdirektoratet [4] in 2021, over 100,000 sheep were lost, and most of these sheep were lambs. Many of the killed sheep are never found, and the cause of death is therefore hard to decide. Although there might be predators living in the rangelands where the sheep are grazing, only around 17% of all of the lost sheep was declared killed by predators in 2021 [6]. The rest of the sheep are assumed to either have died of various types of diseases or accidents [7], or possibly wandered off from the rest of the sheep, making it hard for the collection crew to find them.

1.2 Existing technologies

There are already available technologies which help farmers monitor and retrieve their sheep from the rangelands. The traditional bell is still widely in use, but radio bells and the use of unmanned aerial vehicles (UAVs) has also been employed in recent years to support sheep farmers in monitoring and gathering their flock.

1.2.1 Bells

The traditional way for a farmer to more easily find sheep is to hang bells around their necks. This allows the farmer to hear the sheep when they are close. There are obvious problems with bells, such as the farmer needing to get close enough to hear the bells in order to find the sheep. Another issue is that lambs can not wear bells, as their necks grow too much during the summer season. In spite of these issues, bells are cheap enough to still be a viable option, as the farmer can afford to put them on a larger portion of the sheep.

1.2.2 Radio Bells

A modern approach to the bell is to attach a radio or GPS beacon around the sheep's neck. This allows the farmer to continuously monitor the sheep's location, to see their movements and to easily find them when retrieving the sheep after the summer. The largest providers in the Norwegian market are Smartbjella, Findmy and Telespor [8–10]. While also suffering from the same issue as traditional bells when it comes to lambs not being able to wear them, these bells are far more expensive. The prices for a single bell from Smartbjella, Findmy and Telespor are 999 NOK, 1980 NOK and 989 NOK respectively, according to their websites [8–10]. In addition to the cost of purchase, comes yearly costs per active bell of 149 NOK, 229 NOK and 169 NOK respectively. Given that the yearly profit on a sheep is approximately 600 NOK and given the cost of investment and subscription, most farmers decide to only equip a smaller fraction of their herd with radio bells [11]. These farmers therefore depend on the sheep roaming in groups. A survey done on the use of radio bells found that in average 33% of the sheep were using radio bells [12]. This makes it quite likely that some sheep might still wander off from sheep wearing radio bells. As sheep often graze in areas with poor cellular coverage, problems might also arise while transmitting data back to the farmer. In spite of these drawbacks, these systems are seeing wide adoption among Norwegian sheep farmers, as it makes it easier to both monitor and retrieve their sheep.

1.2.3 UAV

Some farmers are using UAVs to help them find their missing sheep. These UAVs are primarily quad copter drones equipped with cameras. Drones allow the farmer to scout for sheep faster than walking across the vast rangelands. Drawbacks of using these UAVs are that the farmer has to operate them manually while looking for sheep in the video stream transmitted to the controller, and manually make notes of the sheep's locations. Sheep which are partially hidden underneath trees and bushes might be hard to spot in the moment. The same is true for e.g. a brown sheep on brown ground, where the lighting conditions are sub-optimal. In addition to being error-prone, it can be very time consuming and tiring to manually scan for sheep in the images. This UAV approach is seeing wider adoption as it is more efficient than the traditional sheep monitoring, and might be considered a predecessor to the system proposed in this thesis.

1.3 Goal and Research Questions

The objective of this thesis is to assist sheep farmers in overcoming some of the challenges presented in section 1.1. This thesis is part of a project that attempts to locate missing sheep by using object detection networks on images taken by autonomous UAVs equipped with RGB and IR cameras. The proposed system will hopefully assist in monitoring the herd, reduce the number of lost sheep, reduce the number of hours spent by farmers during round-up and increase the sustainability of Norwegian agriculture.

Goal | *"Develop, train and evaluate a model that can accurately and efficiently detect and count sheep using RGB and IR images taken by UAVs."*

The overall goal is to develop a machine learning model that can detect sheep in images. It should be trained and evaluated with real life images from UAVs, using relevant performance metrics. The model could then be utilized in a production system, which will help the sheep farmer in doing weekly counts and to locate sheep during the annual roundup. The system should also reduce spent man-hours by allowing the roundup crew to move in a more targeted manner, by gathering sheep from the areas where they were detected. In addition to the overall goal, the thesis focuses on some more in-depth research questions which will help measure the effectiveness of the proposed solutions.

RQ1 | *"How do IR data usage, tiling and model size variations affect the accuracy of the model?"*

Will a network, modified to also utilize the extra information from the IR images, give better results than one using only the RGB images? If so, is it better to use a fusion network with a separate backbone for each image, or using a linear network with four input channels? Secondly, will the process of tiling the original image yield better results than downscaling and running inference on the original image? Finally, what effects will the selected model size have on the accuracy of the model? The hypothesized answer to this research question, is that the use of IR images, tiling and larger models each will improve accuracy in most or all metrics.

RQ2 | *"How do IR data usage, tiling and model size variations affect the processing time of images?"*

The hardware constraints when using a mobile device, might have an impact on what kind of model variations the device is able to handle. These constraints must be seen in relation to the time constraints posed by the image capturing process. The extra processing required for IR images, tiling and a larger model all affect the processing times. It will therefore be interesting to see how the model variations perform on different hardware. For this research question, the hypothesis is that the all three model improvements will increase the required image processing time. The usage of IR data might require a larger neural network, which generally results in increased inference times. Tiling adds additional preprocessing and postprocessing steps, in addition to requiring multiple inferences per image, all of which drastically increase the processing times. Model size variations also drastically affect processing times, and a hypothesis is that the smaller model will be the reasonable choice for the mobile devices, while the larger model will only be applicable to server-grade hardware.

It can be interesting to evaluate how well the different models detect sheep of varying colors, as well as evaluating the model's performance on sheep that are partially hidden by vegetation or

RQ3 | *"How well can the models detect sheep of varying colors and sheep which are partially hidden by vegetation?"*

image borders. The hypothesized answer to this question is that it will be easier for the models to detect white sheep than other darker colors, and that occluded sheep will generally be much more difficult for models to detect.

1.4 Scope and Limitations

This thesis only addresses the detection of sheep in UAV images and the corresponding evaluation of this process. It does not look into every aspect of a complete production ready application, such as UAV flight routes, autonomously controlled UAVs, the transmitting of images and results, the development of the end-user application etc.

During the research period, there were inhibiting factors which limited the work on this thesis. Firstly, as the development and training of models is time consuming, the number of model variations was limited. Secondly, a drone with better capabilities than the one used within this thesis might be used in during application. The drone used during data capture limits the learning process of the thesis' models by not offering the highest quality image data. Thirdly, the hardware devices used for model inference might not be representative during application, and serve only as a point of reference and comparison for future research. These limitations are all discussed in chapter 7.

Chapter 2

Related Work

In recent years multiple theses and papers have been written on the subject of utilizing UAVs and neural networks to find grazing sheep. In this chapter some of the most relevant are presented.

2.1 Locating Sheep with YOLOv3

Locating Sheep with YOLOv3 is a master's thesis from 2019 written by Jonas Hermansen Muribø [13]. This is the first NTNU thesis in which promising results are achieved using a neural network to analyze UAV images of sheep. The thesis describes experimentation with detecting sheep using subclasses for the sheep's colors or using a general superclass for all sheep, and finds that the superclass solution yielded the best results. The thesis' best results include an mAP0.5 score of 99.1% when validating on the validation set, using images of size 832 px. It is worth noting that the images used in training and validation mostly consisted of grassy fields, and the model would probably not give the same results on a dataset with images of sheep in rougher terrain. This thesis will apply their results by mainly focusing on the superclass detection of sheep.

2.2 Towards Improved Sheep Roundup

Towards Improved Sheep Roundup is a master's thesis from 2020 written by Kari Meling Johannessen [14]. The thesis expands the UAVs dataset with more diverse images of sheep in rough terrain. Johannessen makes an effort to use the extra information captured by the infrared camera on the drone, by proposing a model which fuses the input from the RGB image with the IR image at certain depths in the model. The proposed model consists of a ResNet backbone and a self made head which outputs results as a confidence for cells in a 3×3 grid, inspired by the original YOLO head. As the network is also using tiling, the result for the whole image is a 7×8 grid. As this might suffice for the localization requirement for the task, it hopefully saves time compared to outputting a bounding box for each sheep. The thesis furthermore compares models of different input type, input size, backbone size and fusion depth. Results indicate that using images from both RGB and IR camera fused in the network yields better results than only using the RGB images, though with an increase to inference time compared to simpler models. The best model with fusion achieved 96.3% grid AP on the validation set and 94.5% grid AP on the test set with an inference time of 0.586 seconds on two NVIDIA GeForce GTX 1080. Based on these results, this thesis will strive to develop a fusion network which uses both RGB and IR data, within a state-of-the-art architecture.

2.3 Real-time Sheep Detection

Real-time Sheep Detection is a master's thesis from 2021 written by Ole K. Furseth og Anders O. Granås [15]. The thesis focuses mainly on developing a model that detects sheep in real-time on mobile devices. In order to achieve this they use the smaller model sizes of YOLOv5, and they developed an app for Android phones in order to simulate the model performance on mobile devices. As expected the inference time is affected by the change of runtime environment, but surprisingly the simulations show that the mAP also changes for some of the models. Inspired by Johannessen's results using both the RGB and IR images, the thesis attempts to train models using only MSX IR images, which are IR images with edges superimposed from the RGB image. The thesis shows that these MSX models are not as efficient as the best RGB models. The best RGB model achieved an mAP0.5 of 91.7% on the test set with an inference time of 0.387 on an NVIDIA GeForce GTX 1050m and 1.571 seconds on a Huawei P30 Pro smartphone. In order to properly compare the MSX models to the RGB models, the RGB model can only use RGB images of which there exists a corresponding IR image with MSX. This choice reduces the set of usable data to a small subset of the original dataset, which is then split into training-, validation- and testing datasets. The smaller dataset might have made it difficult for the models to learn, generalize and achieve good results. This thesis continues their work on smaller and faster architectures, by comparing a smaller model's performance to that of a larger one.

2.4 Sheep in UAV images

Sheep in UAV images is a paper from 2021 written by Farah Sarwar, Anthony Griffin, Saeed Ur Rehman and Timotius Pasang [16]. The paper compares different neural network architectures for detecting and counting sheep in images taken by UAVs. The dataset on which the models are trained and evaluated was captured in New Zealand and consists solely of images of sheep in paddocks. U-Net-MS was the best model trained, and achieved great results with a precision score of 98.58% and recall score of 98.02% on the test set with an inference time of 1 second on a NVIDIA GeForce RTX 2080. They also compared whether the U-Net-MS model achieved different results when detecting sheep on images from an altitude of 80m or 120m, as it would be able to cover larger areas when flying at higher altitudes. They found that it performed slightly better at lower altitude, but discussed that this might be due to the images in the 120m dataset contained a higher rate of non-sheep objects than the 80m dataset. In addition to this, they found that a U-Net-MS model trained exclusively on images taken at an altitude of 80m performed very poorly when applied to images taken at 120m, and vice versa. Notably, when trained on both altitudes, the model performed well on both datasets. This thesis will, in inference time requirement estimates, assume a flight altitude of 80m, based on their results. In spite of this decision, the dataset will consist of images of different flight altitudes, in order to make the model robust to height variations.

2.5 Specialization project

This thesis is based on previous work done in the specialization project during the autumn of 2021 [1]. In the project, YOLOv5 is chosen as the architecture to use for training and further development. During the specialization project some preliminary models are developed and trained, although most of the models are trained only on RGB images of sheep, at image size 1280 px. A prototype version of the fusion network is also trained using both RGB and IR images. Although the results are promising, it must be noted that the models are only evaluated on the validation set and not on a separate test dataset, making the results less trustworthy. During the specialization project, it was attempted to apply pre-trained weights and biases to a model, in order to possibly increase performance and decrease training times when training on UAV images of sheep. Although the training times were reduced, the model performance remained about the same, and there was no clear indication that pre-trained weights improved the model compared to a model trained from scratch. Based on this, this thesis will train all models from scratch.

2.6 Chapter summary

There are many related theses and papers on the subject of detecting sheep with UAVs. Many of them have achieved great results, but it is hard to compare the models itself against each other when the datasets are completely different. However, it is possible to learn from the methods utilized in order to improve the results. This thesis builds upon the work of Muribø [13], Johannessen [14], Furseth and Granås [15] and Sarwar et. al [16], in order to both improve upon their work and attempt new techniques, all in an effort to simplify the work for sheep farmers.

Chapter 3

Theory

This third chapter presents theory about neural networks and their applications, important features of a good dataset for training such networks, relevant metrics for measuring the performance of a neural network and specializations of these topics relevant to the thesis.

3.1 Neural networks

Neural Networks (NN) is a branch of machine learning computer science, in which structures simulating those found in the brain are applied to solve complex problems. Networks are built up of units called *neurons*, and each neuron holds a value, which depends on its connection to neighboring neurons [17]. The purpose of neural networks, is to “train” a model to solve problems on its own, without explicitly programming the connection between input- and output values. One method of training can be described as feeding the network a set of input values and desired output values, and allowing the network to iteratively adapt itself until it reaches the desired result. This method of training is generally called *supervised learning*, due to the method’s requirement for a human-given ground-truth for the network to compare its results [18].

Figure 3.1 is an example which shows the three sections of which a traditional neural network consists. Each circle in the figure represents a single neuron, and each column is referred to as a layer. On the left, in blue, lie what are called the input neurons or the input layer. In e.g. image processing, this layer represents the entirety of the image, and each neuron represents the value of one color channel (red, green or blue) in a single pixel of the image. In the middle of Figure 3.1, in green, lie what we refer to as *hidden layers*. These layers allow the model to evolve and behave in a more complex manner. A higher number of these layers will increase the amount of learning the network can take in, but also drastically increase the time required to train the network. On the right, in light blue, is the output layer which can look very differently depending on the function of the network. A classic example of a neural network is one that reads images of hand written digits, and outputs which digit is in the image. In such an example, the network output would probably consist of ten neurons representing the digits 0 through 9.

As information runs through a neural network, it moves as in a flow diagram [17]. The values move from input towards output (left to right in Figure 3.1), and are affected by two “settings” in the neural network. Each “connection” contains a setting called a *weight*, and every neuron is defined by a setting called *bias*. When a value flows from a neuron into a connection, the value is multiplied by the connection’s weight. The value of a neuron (except the input neurons) is defined as the sum of the values of connections pointing into the neuron, plus the value of the neuron’s bias. This value is then propagated deeper into the network, until every output neuron has received its value [19].

When training a neural network, the weights and biases are changed by a *back propagation* algorithm [20]. The changes are based on how large the loss of the previous batch was, where *loss* is the difference between the predicted result and the ground truth result given by the human. A *batch*

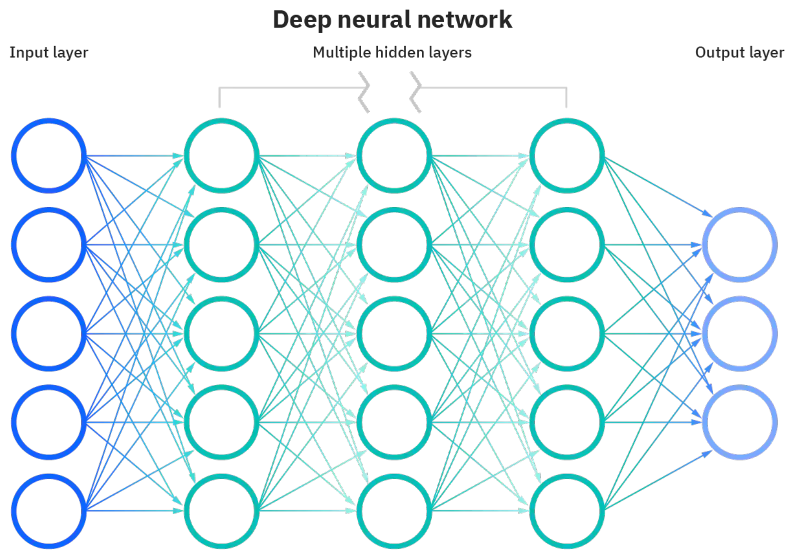


Figure 3.1: An example of the structure of a neural network [18].

is a subset of the training data which is processed through the network before model parameters are updated during back propagation. Many neural networks also calculate the current model's fitness. *Fitness* can be crudely described as the inverse of the loss, meaning how well the model performed, calculated only at the end of an *epoch*. An epoch is a complete iteration through the whole training dataset. By making notes of the fitness after every training epoch, the network can begin to see when its training is no longer making progress. The *patience* is a parameter that decides how many epochs the network keeps training without seeing an improvement before concluding the training process [21].

The term *model* generally refers to a combination of the structure of the network, as well as the current weights and biases. When storing a large machine learning model, these weights and biases take up most of the storage space. The network in Figure 3.1 is a *fully connected neural network*, meaning every neuron in one layer is connected to every neuron in the previous and the next layer. For large models, this design is not feasible, as it requires impossible amounts of memory and processing power. E.g. in cases like image processing, where the input image might have a size of 10 megapixels, there are ten million input neurons. To connect these to a neighboring layer of equal size, the model would need 10^{14} ($O(n^2)$) connections. This amount of parameters is not realistic for even modern computers when training the model, nor when storing the model.

3.2 Convolutional neural networks

Convolutional Neural Network (CNN) is a term which describes neural networks utilizing convolutional layers, making them excel at image processing [22]. These networks solve the issue that arises with large, fully connected networks by not connecting every neuron, but rather looking at each layer of neurons as an image and only connecting each pixel to its neighboring pixels in the next layer. This design reduces the required connections from $O(n^2)$ to $O(n)$.

In Figure 3.2 two layers are visualized, and between them a *convolutional kernel*, which is the cornerstone of all CNNs. This $x \times x$ kernel calculates the value of a pixel in a layer, from x^2 pixels in the previous layer. This requires $O(x^2 \times n)$ connections between each layer, but because the kernel is used for an entire layer, it's only necessary to store $x \times x$ weights which will be re-used for every pixel in the layer. This concept is called *shared weights* and saves large amounts of storage and memory use, while still offering good results [23].

Intuitively, a convolution can be thought of as a filter that looks at a small piece of the image at the time. Using Figure 3.2 as an example, the nine pixels in the kernel describe what sort of pattern should be recognized in the source layer. If the kernel is used on a matching pattern, it yields a high output, otherwise it yields a low output.

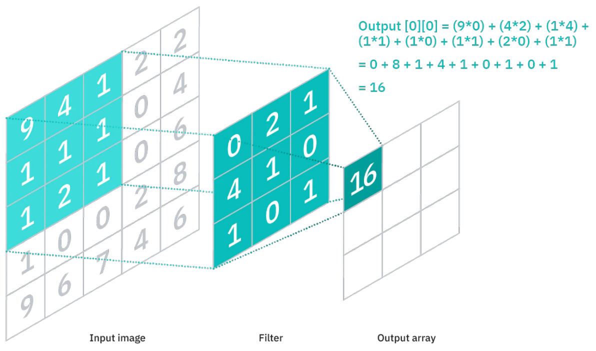

Output [0][0] = (9*0) + (4*2) + (1*4) + (1*1) + (1*0) + (1*1) + (2*0) + (1*1)
= 0 + 8 + 1 + 4 + 1 + 0 + 1 + 0 + 1
= 16
Input image
Filter
Output array

Figure 3.2: Example structural diagram of a CNN [22].

3.3 Object detection networks

Object detection networks are most often built upon a CNN architecture, and are used to locate and classify multiple *classes* of objects in images [24, 25]. Typical examples of such classes are annotations such as “house”, “person” and “dog”. A network’s goal is to take an image as input, and return a list of *bounding boxes* for the objects, as well as the correct label for what each object is. The network manages this by recognizing constellations of landmarks in the images, which are defining for each class. Applying a fully trained network to locate objects in previously unseen images is often called *inference*.

3.3.1 YOLOv5

YOLOv5 is a state-of-the-art object detection network [26]. It is open source, developed and managed by Ultralytics, with most contributions attributed to Glenn Jocher. The network shows very good results on the Microsoft COCO dataset, both with regards to accuracy and inference times [27]. There are still no published papers presenting YOLOv5 specifically, but the architecture is actually the same that is used in YOLOv4 [28], although some changes have been made to improve the training process [29]. Another notable difference is that YOLOv5 is implemented in Python using PyTorch, while YOLOv4 is implemented in C using the Darknet framework [28].

YOLOv5 consists of three main sections, as seen in Figure 3.3. The network’s backbone consists of a CSPDarknet53 [28], which is known to be both fast and accurate. The backbone is responsible for fetching basic features like edges, color, etc. from the image. In addition to this, the Spatial Pyramid Pooling (SPP) is used to highlight the most important features, with almost no extra time consumption. The neck consists of a Path Aggregation Network (PANet) [28], in which the information in the detected features is further processed. From the backbone to the neck are also some skip paths, namely Cross Stage Partial connections (CSP) [28]. The head depends on the same anchor based architecture as found in YOLOv3 [30]. In the head, detected objects in the image are represented as a combination of class labels, bounding boxes and confidences.

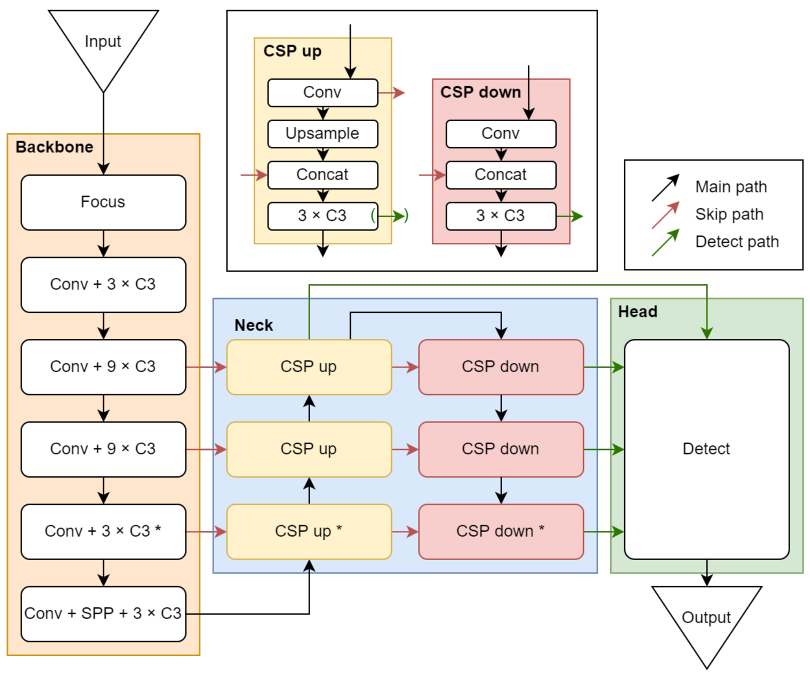


Figure 3.3: The architecture of YOLOv5, divided into three sections. Boxes marked with an asterisk (*) are not present in the YOLOv5s architecture, but only present in the YOLOv5l6 architecture.

3.4 Dataset

As previously mentioned, object detection networks are trained using supervised learning, meaning the network is given a large number of images with corresponding labeled bounding boxes. The network attempts to predict where the boxes are, and what classes they represent, and then adapts to yield improved results next epoch.

Glenn Jocher, main contributor to YOLOv5, points out that in order for a network to be trained effectively, it is important that the given dataset possesses certain properties [31]:

- The dataset must contain many examples of each class of object. If a class is under represented in the training dataset, the network will never learn to recognize it. A larger representation of the class will yield a more general class “definition” within the network, and it will more easily recognize instances of the class in new images.
- The dataset must be varied. If all images are similar, and the network is tasked with recognizing objects in images different from the training data, it will not give good results. By training on a varied dataset, the class definitions will be more generalized and yield better results.
- The dataset must be large. This follows from the two previous points, but in order to maintain a sufficiently varied dataset with a sufficient number of examples of each class, a dataset should probably consist of several thousand images.

3.5 Augmentation

Convolutional neural networks show good results compared to other approaches, but are limited by the amount of training data they require in order to avoid overfitting. *Overfitting* refers to a network learning too specific and non-generalized patterns from the training dataset that can not be applied to a test dataset or an actual use case. One solution to this problem would be to gather more data, but for many use cases, e.g. the one presented in this thesis, this task is quite time consuming. Another way to solve the lack of data, is to utilize data augmentation [32].

The point of data augmentation, as stated above, is to generate more training data from the already available data. YOLOv5 augments the data using multiple methods, including color space manipulation, transforming and most noticeably - mosaic augmentation [33]. These augmented data can be seen in Figure 3.4 where the most visible effect is the mosaic augmentation which cuts multiple images into pieces, rotates and scales them, and then combines them to mosaics.

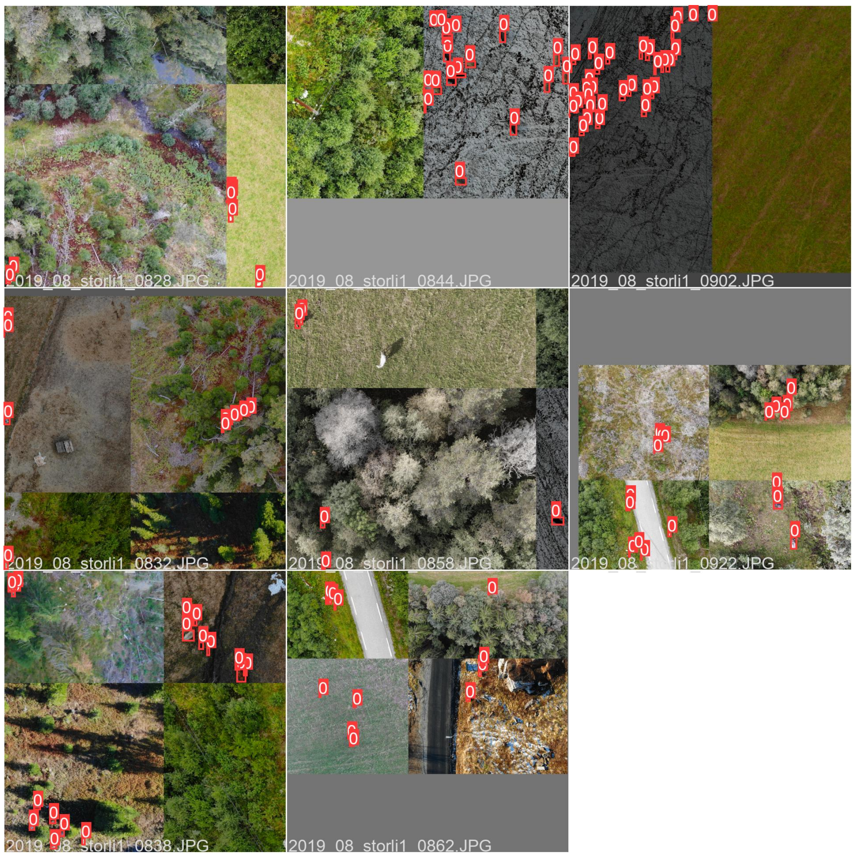


Figure 3.4: Augmented training data as input to YOLOv5 during a training with a batch size of eight.

3.6 Image sizes and tiling

When a network is tasked with finding objects in images, its performance depends on the objects being large enough, i.e. covering enough pixels, for the network to recognize the defining landmarks for the object's class. As previously mentioned, a network takes a given number of neurons (pixels) as input. If a network is built to take in large sized images, issues arise with increased memory usage [34]. Because of these issues, many neural networks crop or scale down the input-image, for then to train on those images. As an example, if one attempts to train a network designed with 1000×1000 pixels (1 megapixel) as input by inputting an image of size 4000×3000 (12 megapixels), then 11MP/12MP of data will be lost, either by parts of the image being completely cropped out and missed or by losing small details in the image as it is scaled down by a factor of 12.

A possible solution to this problem is *tiling*, which is the process in which large images are cut into sub-images (tiles) that fit perfectly into the neural network [34, 35]. The tiling process ensures that no data is lost, by avoiding the cropping/down scaling process entirely. Tiles are laid with a notable amount of overlap, to ensure that all objects that are whole in the original image are also whole in at least one image tile [14]. This is important as neural networks often struggle to recognize parts of objects. Therefore, by ensuring every object is kept uncut in at least one tile, the network will find them much more easily [36].

3.7 Fusion in object detection networks

Johannessen [14] developed a neural network in her master's thesis based on ResNet, which accepts a pair of images, namely one ordinary RGB image and one infrared image. The network processes the images through two separate backbones, before fusing the data together and leading the combined data through the remaining part of the network. The term *fusion* denotes this process of combining the data from two parallel sub-nets, and the purpose of this fusion is for the network to "learn" more from the pair of images than it could from a single RGB image. Both Johannessen and Ophoff et. al [37] claim that another important advantage of the fusion concept is that it's fairly easy to reuse pretrained weights and biases from models using the original architecture through transfer learning. As visualized in Figure 3.5, the fusion step itself is done by "concatenating" the input data, then running a 1×1 convolution [37]. The concatenation layer stacks the two inputs after each other, which doubles the depth. The convolution layer is then used to reduce this depth, such that it fits with the rest of the network.

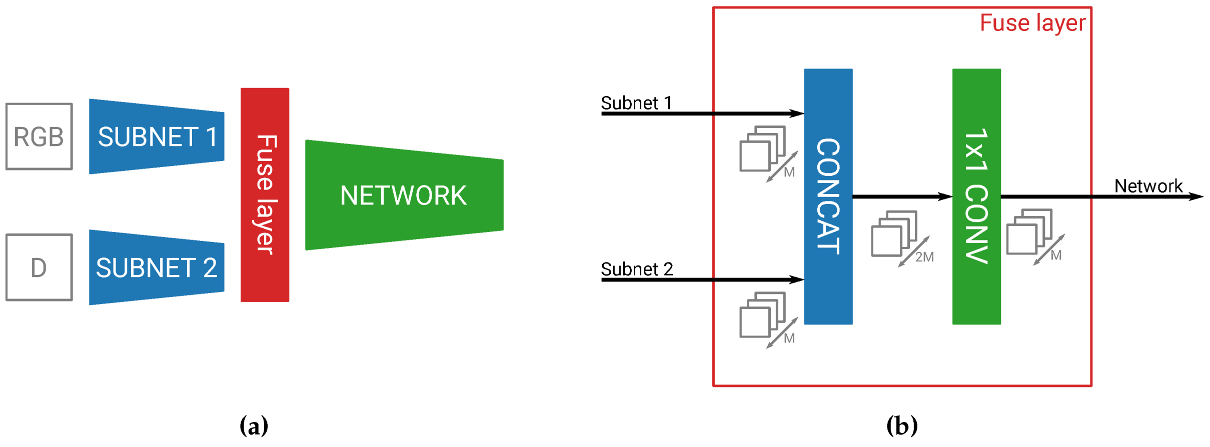


Figure 3.5: T. Ophoff, K. vBeeck, T Goedemés [37] figure for a fusion network architecture **(a)**, as well as the structure of the fusion layer **(b)**.

3.8 Metrics

When comparing different object detection models, it is important to have a set of predefined metrics in order to properly differentiate the models' performance. Padilla et. al [38] discusses the advantages and issues of different object detection metrics, and their work is used as the foundation for this section. The most common metrics for object detection models are Precision, Recall, Average Precision (AP) and mean Average Precision (mAP). The Microsoft COCO competition uses the popular mAP0.5 and mA0.5:0.95 metrics [39]. All the aforementioned metrics are based on the statistical concepts true and false positives and negatives, which in object detection depend on intersection over union.

3.8.1 Intersection over union

After a model has run inference on an image, it returns a set of predicted labels and with the corresponding set of ground truth labels it is possible to evaluate the predictions. When comparing a prediction with a ground truth label, the *intersection over the union* (IoU) of the two labels' areas is

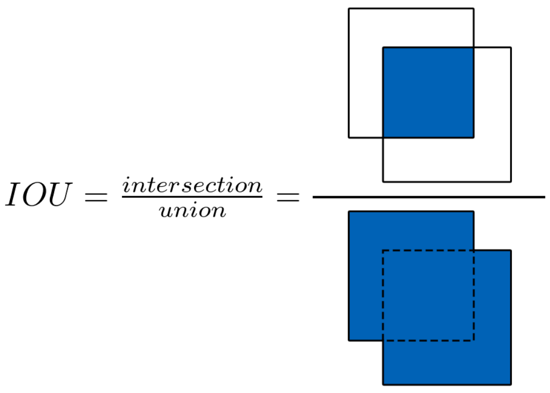


Figure 3.6: Intersection over Union (IoU).

computed, as seen in Figure 3.6. If the computed IoU is exactly 1, the prediction is a perfect match. On the other hand if the IoU is 0, the two labels do not overlap at all [38].

Comparing a prediction's label-IoU with a given IoU threshold, typically 0.5, puts the prediction in one of four categories, which are described below. Three of these categories are shown in Figure 3.7.

- True Positive (TP): If the computed IoU of the label pair is greater than the threshold.
- False Positive (FP): If a detection label does not have a ground truth label above the IoU threshold.
- False Negative (FN): If a ground truth label does not have a matching prediction label.
- True Negative (TN): If the background is not predicted. This category is not used for bounding box metrics.

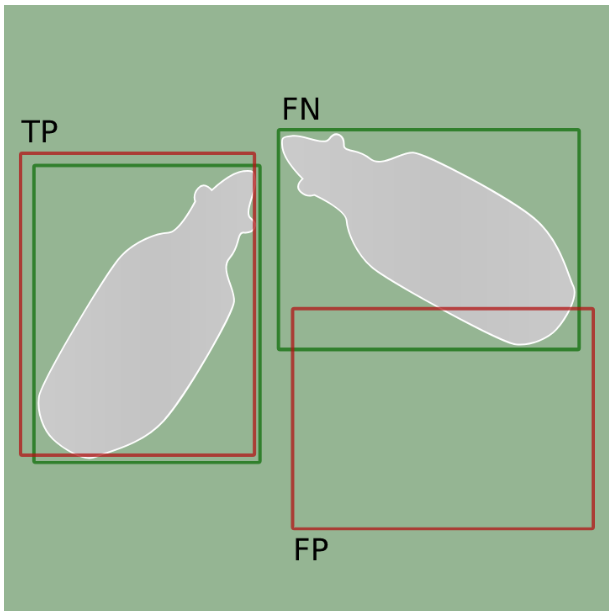


Figure 3.7: Example showing True Positive, False Positive and False Negative. Green boxes represent ground truths and red boxes represent model predictions.

3.8.2 Precision and recall

Precision and recall are two of the simplest and most useful metrics used when attempting to understand a machine learning model's prediction performance. *Precision* is computed using the formula found in Equation 3.1, and represents the portion of all predictions which are correct. *Recall* is computed using the formula found in Equation 3.2. Put into words, it represent the portion of the labeled objects the model finds. Figure 3.8 shows examples of these metrics [38].

$$Precision = \frac{TP}{TP+FP} = \frac{correct\ predictions}{all\ predictions} \tag{3.1}$$

$$Recall = \frac{TP}{TP+FN} = \frac{correct\ predictions}{all\ ground\ truths} \tag{3.2}$$

Both of these metrics are relatively easy to understand, but do also have some disadvantages. Precision will still achieve a perfect score, even if it predicts too few objects, as seen in **(a)** in Figure 3.8. Recall, however, will achieve a perfect score even though the model predicts too many detections, as seen in **(c)** in Figure 3.8. In order to effectively describe a perfect score for both precision and recall, as shown in **(b)** in Figure 3.8, an improved metric that combines the scores from both precision and recall is needed [38].

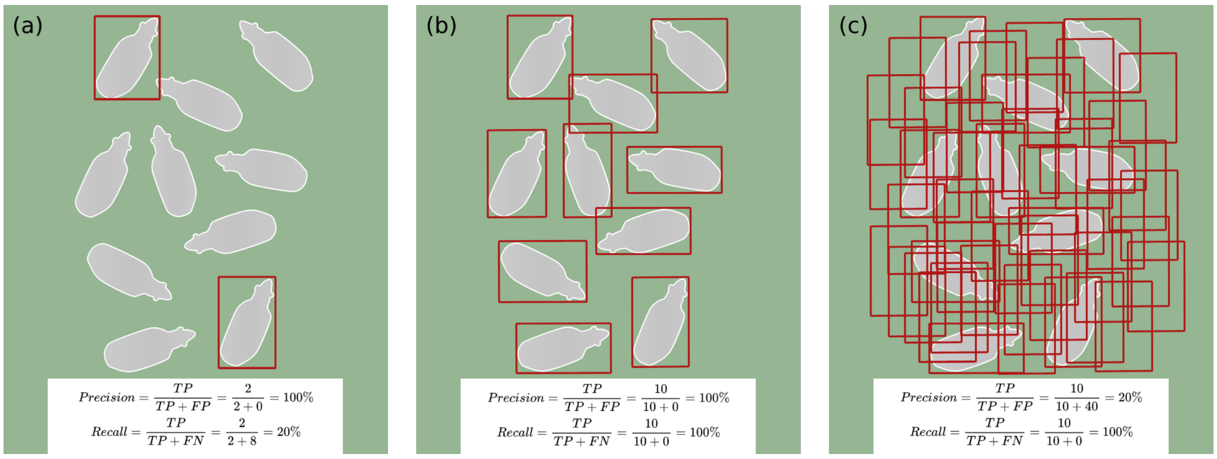

(a)
$Precision = \frac{TP}{TP+FP} = \frac{2}{2+0} = 100\%$
$Recall = \frac{TP}{TP+FN} = \frac{2}{2+8} = 20\%$
(b)
$Precision = \frac{TP}{TP+FP} = \frac{10}{10+0} = 100\%$
$Recall = \frac{TP}{TP+FN} = \frac{10}{10+0} = 100\%$
(c)
$Precision = \frac{TP}{TP+FP} = \frac{10}{10+40} = 20\%$
$Recall = \frac{TP}{TP+FN} = \frac{10}{10+0} = 100\%$

Figure 3.8: Example of precision and recall edge cases.

One method to combine precision and recall is using the F1-score formula, as seen in Equation 3.3. The *F1-score* is defined as the harmonic mean between the precision and recall. As all predictions are given with a certain confidence, this function is used to find a confidence threshold which maximizes precision and recall in combination. This threshold is chosen by calculating the F1-score for all confidence thresholds, and then using the confidence threshold which got the highest result. The F1-curve is a graph showing the relation between a confidence threshold and an F1 score [38].

$$F1 = 2 * \frac{precision * recall}{precision + recall} \tag{3.3}$$

3.8.3 Mean average precision

Another improved, and often used metric for object detection, is called *mean average precision* (mAP). mAP is calculated by finding the curve for precision and recall by using confidence thresholds, i.e. if a prediction is below the given threshold, it will be ignored [38]. Then by plotting the $Pr \times Rc$ curve with confidences and calculating the area under the curve (AUC), we find the metric called Average

Precision. An example of the $Pr \times Rc$ curve can be seen in Figure 3.9. Unfortunately the $Pr \times Rc$ curve is in most cases zigzag-like which makes it challenging to accurately measure the AUC. Therefore N-Points Interpolation or All-Points Interpolation is used for the computation of the Riemann integral. If the model predicts multiple classes it can be reasonable to use a single metric combining AP from all classes. This is done by computing the average of the AP over all classes, which is called mean average precision [40]. The Microsoft COCO competition metric mAP0.5 is the mAP with an IoU threshold of 0.5 using N-Points Interpolation with $N = 101$ [38]. AP0.5:0.95 expands this metric by computing the AP0.5, AP0.55, AP0.60 etc. up to AP0.95, then taking the average of results from all the different IoU thresholds.

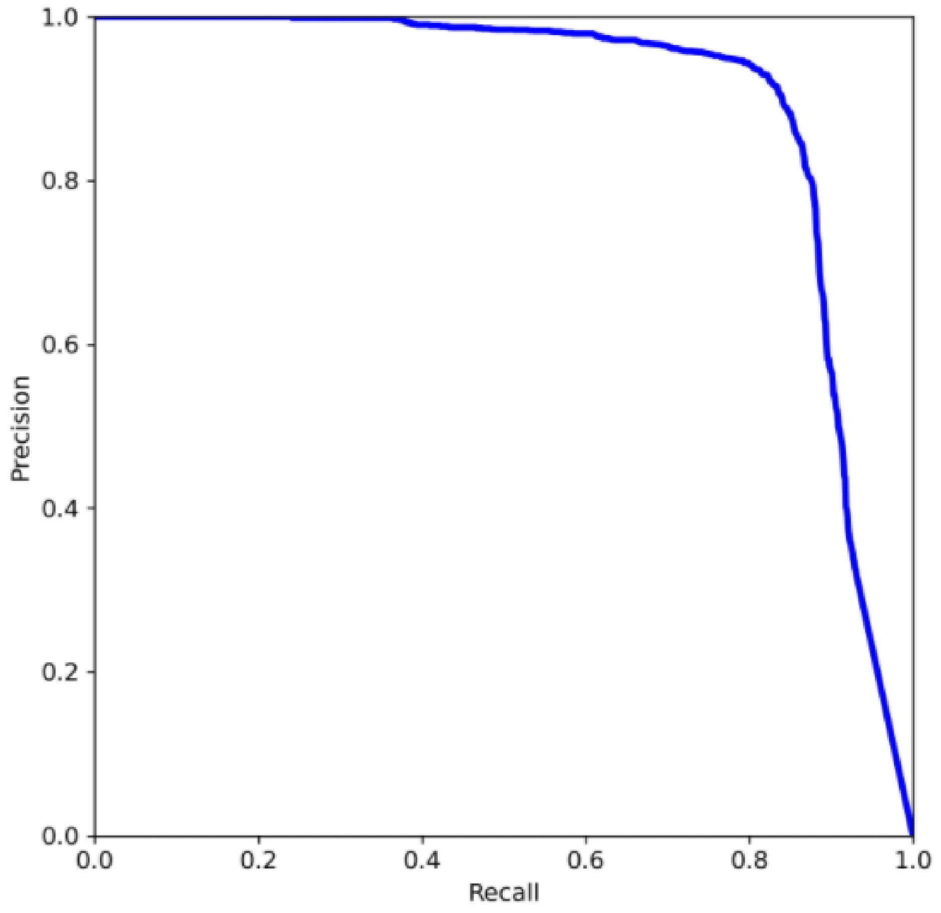


Figure 3.9: Precision recall curve, showing the relationship between the two properties.

3.8.4 Grid metrics

In the object detection network developed in Johannessen's thesis [14], the network head does not predict traditional bounding boxes of sheep. The network instead divides the image into an 8 × 7 grid, and then predicts whether there is at least one sheep within the bounds of each grid cell. When labeling the data, Johannessen labels each sheep with a bounding box, to simply make sure each sheep is accounted for. These bounding boxes are then transformed into grid based ground truths, as shown in Figure 3.10, where each grid cell is described with the colors green, blue and gray, respectively representing the categories *true*, *false* and *ignore*.

Johannessen designed a set of rules to decide which category and color each cell falls within. The first rule is that if at least one bounding box overlaps with a grid cell, where over 20% of the bounding

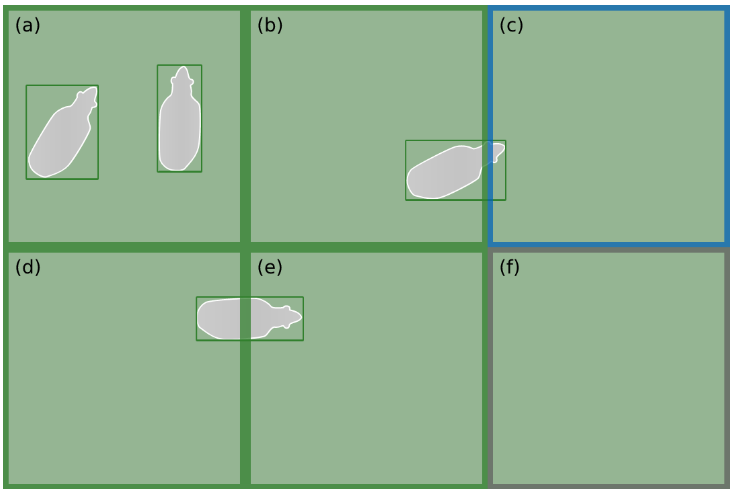


Figure 3.10: Example of precision and recall edge cases.

box' area is within the cell, then the cell is categorized as true and colored green. The second rule is that if there are no sheep bounding boxes overlapping with the grid cell, then it is categorized as false and colored gray. The third rule catches the remaining bounding boxes, so that every cell that (only) overlaps with sheep bounding boxes where the overlap is less than 20% of the bounding box area, is marked to be ignored, signified by the blue border color. A property of Johannessen's ignored cells is that it no longer matters whether the network detects the slightly overlapping sheep or not. Whether the network does detect these overlaps depends on how the sheep in question collides with the grid cell, but by ignoring all cells that barely contain a sheep, the network is trained on more relevant data. So whether the network detects the partial sheep or not, it is not punished nor credited in any case. It is worth noting that the network itself only has two categories for each cell, meaning it could only predict a true or false cell.

When calculating the efficiency of her models, Johannessen utilizes four grid-related metrics. The first two metrics used are grid precision and grid recall, which work in a similar way as ordinary precision and recall, as described in subsection 3.8.2. By initially filtering out the ignored cells, grid precision and grid recall are calculated by first counting each grid cell as a true or false positive or negative, then counting up for each category and finally calculating using the formulas found in Equation 3.1 and Equation 3.2. The third metric utilized by Johannessen is called sheep recall, which describes how many of the total sheep are discovered. Sheep recall is calculated by counting each bounding box that is covered by a predicted true grid cell and dividing by the total count of bounding boxes in the ground truth. The fourth, final metric Johannessen uses is a variation of average precision, using the grid predictions as input. There are obviously no IoU-thresholds necessary as the grids always align perfectly, and otherwise the metric works in the same way as ordinary mAP, as described in subsection 3.8.3 [14].

3.9 Non-maximum suppression

Non-maximum suppression (NMS) is an algorithm that runs within the last step of an object detection network's detection. The network first decides to place a bounding box over any pattern that matches the pattern it has learned, which often results in multiple overlapping boxes on each object. To remedy this, the non-maximum suppression algorithm is run, to suppress the non-most-confident bounding boxes, i.e. to only show the ones most likely to be correct. An example can be seen in Figure 3.11.

To explain the algorithm in short, it begins by picking out the most confident bounding box in the

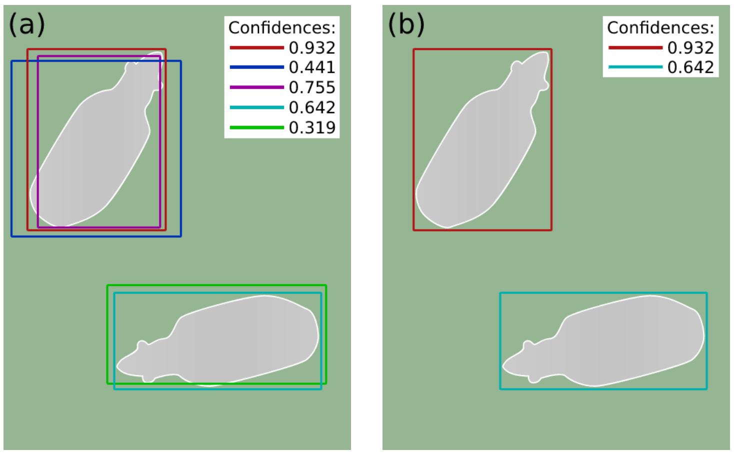


Figure 3.11: Example run of the NMS algorithm, where (a) shows all bounding boxes before the suppression, and (b) shows the remaining bounding boxes after suppression.

image, and storing it. It then compares it with all other bounding boxes, and if the IoU between them is higher than a given IoU threshold (defaults to 0.45 in YOLOv5), then the second bounding box is discarded. Once every box has been compared with the highest confidence one, the algorithm starts over, and stores the next highest confidence bounding box, then compares every other remaining box with it. This goes on until there are no more boxes available. Finally, the stored boxes are returned [41].

3.10 Statistical significance and the null hypothesis

Statistical significance is a term used to describe the certainty required for a scientific study to define something as true, based on a null hypothesis and a given significance level α [42]. The null hypothesis states that there is no difference between the properties of the tested entity compared to the control group. The significance level describes the required probability of the null hypothesis being wrong before it is discarded, with a certainty of $100\% - \alpha$. Thus if the probability of a result occurring is lower than α when assuming that the null hypothesis is correct, then the null hypothesis is discarded.

By describing the outcomes of a stochastic process as a normal distribution, the statistical significance can be described using the distribution's standard deviations. For many probability distributions, including the normal distribution, the probability of the process producing a certain value is directly tied to the standard deviation [43]. Figure 3.12 shows a normal distribution, with some integer standard deviations marked off. The *empirical rule* states that 68.3% of values will end up within ±1 standard deviation (σ) of the mean (μ). In a similar fashion, 95.5% and 99.7% of values will end up within $\mu \pm 2\sigma$ and $\mu \pm 3\sigma$, respectively [44]. These probability values are all visible in Figure 3.12.

3.11 Linear field of view

The distance covered on a flat surface by e.g. a camera, can be referred to as the camera's linear field of view. By assuming a minimal field of view angle of θ, and a distance between camera and surface of height, h, and the distance of width, w, can be calculated, as seen in Equation 3.4 [45]. $\frac{\theta}{2}$ and $\frac{w}{2}$ are used as angle and width during calculation due to the tangent function's requirement of a right angle. This is better visualized in Figure 3.13.

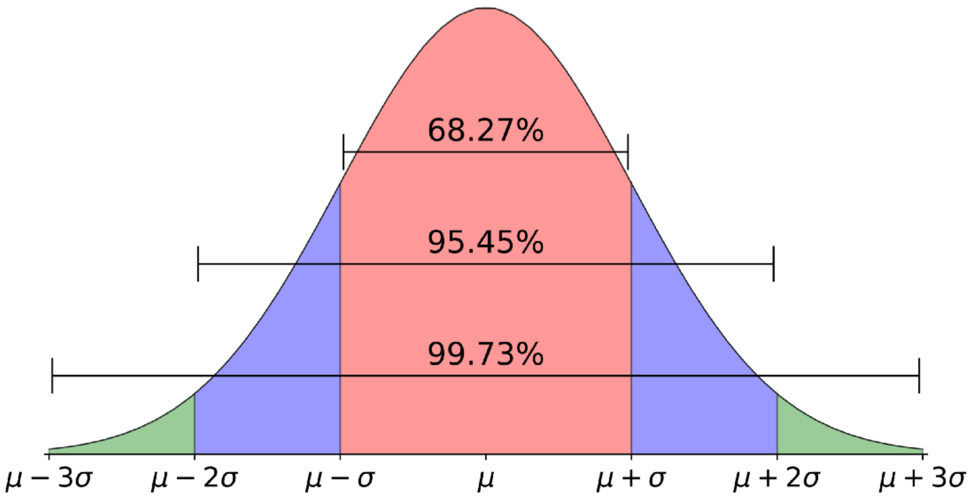


Figure 3.12: A normal distribution, divided by integer standard deviations from the mean, and showing distributional percentages for each section [44].

$$\begin{aligned} \frac{\frac{w}{2}}{h} &= tan(\frac{\theta}{2}) \\ w &= 2 \times h \times tan(\frac{\theta}{2}) \end{aligned} \tag{3.4}$$

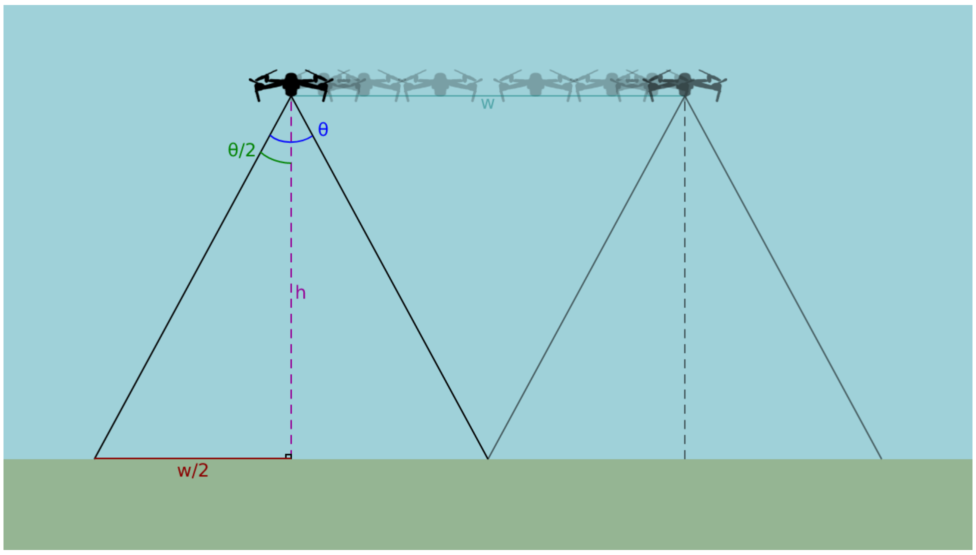


Figure 3.13: Visualization of relevant variables for calculating the linear field of view.

3.12 Chapter summary

Neural networks are built of multiple layers of neurons with connections between them. Each neuron holds a bias parameter, and each connection has a given weight parameter. A trained neural network, consisting of the network structure and the parameters, is generally referred to as a model. When training a model, the back propagation is run once every batch. The back propagation is the process in which the model's parameters are updated in order for the model to perform better. When the network has gone through all the training data, one epoch has passed and the model is validated against the validation set. In object detection it is normal to use a convolutional neural network, which only connects a neuron to its neighboring neurons, in order to reduce the complexity of the network. The task of an object detection network is to find specified classes, which are often predicted as bounding boxes. YOLOv5 is a state-of-the-art object detection network architecture which achieves good accuracy and inference time on the Microsoft COCO dataset. When training a model it is important that the training dataset contains both varied data and large amounts of data, in order for the network to be general enough to recognize objects in new environments. Augmentation is the process of altering already collected data in order to make the dataset more diverse and avoid overfitting, and is used by most modern machine learning networks. Tiling is an optional preprocessing step which cuts the original image into smaller sub-images in order avoid down-scaling, potentially making the model more accurate. Fusion is the process of combining RGB and IR images through two sub nets, in order to be able to utilize more data. The object detection metric precision is the portion of predictions that are correct, while the recall is the portion of labeled objects the model finds. Average precision is an often used metric which is a combination of both the precision and the recall, and is a more useful metric for a model's accuracy. Grid metrics are based on the grid predicting system proposed by Johannessen, and include all the aforementioned metrics in the grid system. Non-maximum suppression is a post-processing step which only keeps the bounding boxes with highest confidence. In a normal distribution 95,5% all values will end up within ± 2 standard deviations of the mean, and limits such as that can be used to decide a result's statistical significance. The linear field is the length of the area covered by a camera's lens, and can be calculated by knowing the height from the plane and the field of view angle.

Chapter 4

Project Description

This chapter presents the project's use cases and some of the possible solutions to the challenges, as well as the requirements such a solution will have to meet. The physical restrictions will be explained, and finally the used hardware- and software products will be discussed.

4.1 Use cases

This thesis focuses on two use cases with challenges to solve, regarding the husbandry of sheep in rangelands. The first use case is to assist the farmers to perform weekly inspections of the sheep during the grazing period, as required by the authorities [5]. These weekly inspections include counting the sheep herd, and are often performed as a collaboration between farmers in the same area, where they take turns doing the inspections. The second use case is to assist the farmers to locate and retrieve the sheep at the end of the grazing period, typically in September [3]. This is a time-consuming task and the gathering of the sheep is divided into three phases [46]. The first main roundup commonly runs over 1-2 weekends, and approximately 90% of the sheep are found during these weekends. This main roundup is often performed in cooperation between multiple farmers and their family and friends. The second main roundup often lasts a several weekends, but have reduced manpower compared to the first roundup. During this phase the farmer typically finds another 5% to 10% of the sheep. The third phase lasts about a month, and the farmer searches for the remaining sheep in a larger area of the rangelands. Some of the straggling sheep might have wandered outside the typical grazing area or died to predators or disease [7], thus this phase can be very time consuming for the farmer who often searches alone.

4.2 Possible solutions

This thesis considers possible solutions to assist the sheep farmers in monitoring and retrieving their sheep. The purpose of the proposed system is to utilize an autonomously operated UAV to capture both RGB and IR images of grazing areas, in search of sheep. The images are automatically preprocessed and fed as input into a YOLOv5 neural network, which attempts to detect sheep, and draw bounding boxes around them. The farmer receives the data as coordinates on a map in an application, and will then know where the sheep are located. At the end of the grazing season, the farmer needs to retrieve the sheep using information provided by the system. In this case, this information is considered timely, which means that the information loses its value after a short while as the sheep might move. Because of this timeliness, solutions must be able to output near-real-time sheep locations.

The proposed solutions differentiate themselves from existing technologies in a few ways. One difference is that these solutions are better suited to locate straggling sheep during the third phase of sheep roundup. Existing radio bell solutions are usually applied to around 1 out of 3 sheep, meaning there is a chance that some sheep or smaller groups roaming about without a single bell among them [12]. The autonomous UAV solution proposed in this thesis, however, does not require the sheep to wear anything, and effectively locates even single sheep. Another advantage is that the proposed solutions can find lambs at the same rate as adult sheep, while radio bell solutions can only be hung around the neck of full grown sheep. An issue, however, is that it is impossible for the drone to detect sheep which are located below trees or otherwise hidden from the camera on the drone. It is also worth noting that for a farmer, the investment into a suitable drone would be dramatically lower than the investment to buy radio bells for a large portion of their sheep. Considering all these aspects, it might be worth considering a hybrid solution where the radio bells are used as today, and the UAV-based solution can be used to locate lost sheep.

There are two solutions within the scope of this thesis. The first solution is a mobile system depending on a smaller machine learning model to perform real time detection on either a phone, tablet, or a portable computer. The second solution is a cloud based system using a larger model. This solution either detects sheep in near-real-time by transmitting the images directly from the drone to the server, or performs all the detections after the drone has finished surveying. The variations between these two solutions are discussed below.

4.2.1 Mobile solution

A mobile solution has been previously discussed in Furseth and Granås' master thesis [15]. They discussed the usage of a smaller version of YOLOv5, which gave acceptable results when ported onto a mobile device. An alternative to their particular mobile solution could be to have the operator bring specialized hardware with them, e.g. a portable computer such as a laptop with a dedicated GPU, to perform detection on relayed data. These mobile solutions have the advantage of saving on mobile traffic, compared to the cloud based solution, but might give poorer results or slower inference times due to the lower quality hardware.

4.2.2 Cloud based solution

For a cloud based solution, the image processing and sheep detection is done on a cloud server. This solution allows for a larger neural network, which generally lets the network learn more and produce better results. Still, this presents a problem regarding the data-transfer from the client to the server. One approach is to simply use cellular network to upload data from the drone directly, but this might be sub-optimal as reception can be poor in highland graze areas. Another approach is for the drone operator to relay the data from the drone using an intermediary device. This would allow the operator to set up in a location where the reception is better, while still allowing data transfer to work well. A problem with these approaches is that it might be quite costly to send high resolution images over cellular network. A third approach might therefore be to upload all images through a wired connection, after the drone has finished its surveying of the area and then start to process them and detect sheep.

4.3 Alternative solutions

The challenges posed in section 4.1 have more solutions than those discussed within the bounds of this thesis. One example of such a solution is to use modern RFID tags on each sheep, and detecting these with a drone carrying an RFID reader. By using active RFID tags, which allows for longer reading distances, locating and counting sheep would be a trivial task compared to analyzing images taken by the drone. These RFID tags cost much less than radio bells, which would allow a farmer to tag each sheep in their flock. In turn, this would reduce the risk of losing wandering sheep. The largest cost to the farmer would be the investment into the drone, although the ownership of the drone could be outsourced to a service provider or handled as a co-ownership between multiple farmers. A promising feature of such a solution, is that it might show improved results on sheep that are visually occluded to the drone, as the RFID signal would generally pass through leaves and branches. There already exists research into solutions combining RFID technology and drone technology, e.g. Buffi et al. [47] and Freed et al. [48]. This RFID-based solution is an exciting possibility, but falls outside the scope of this thesis, which concerns itself with the application of modern image processing to solve the challenges.

4.4 Requirements

Three requirements are defined for the proposed solutions, namely measurable performance, processing times, and model generality. These requirements serve the purpose of focusing the work process, in order to increase development speed and define important aspects of a possible solution.

R1: Measurable performance

It can be argued that during sheep roundup, it is sufficient for the proposed system to locate just one sheep per image, even if there are more than one sheep in the image. The farmer will have to travel to the sheep's location, no matter how many sheep there are. There they will find the rest of the group even if only one is detected by the drone. This point does not hold up in the case of the weekly inspections. In this case, the point is to achieve a relatively accurate count of the sheep, in addition to their locations. Therefore, it is crucial that the proposed solutions are able to accurately find each sheep within the images, to ensure that the weekly counts are precise. Additionally, this thesis aims to measure how accurately the models can actually find each individual sheep. This point is of a more comparative nature, but in order to effectively compare the performance of different models, detailed predictions must be made. This thesis therefore uses bounding boxes when predicting sheep, which allows established metrics for object detection to be used. These metrics are initially precision and recall, as well as the more complex average precision which combines both of them into one metric, all of which are described in section 3.8.

R2: Processing time

For the use cases, especially for the localization of sheep before retrieval, it is imperative that the system manages to complete the image processing before the next image is captured. This is important to avoid increasing queue times, and to avoid having to wait for the results. The time spent moving the drone from one capture location to the next can be estimated, and will serve as the upper limit to the total processing time. The estimation of this time interval is performed in section 5.7.

R3: Model generality

It is important that the trained model is adaptable to variations in the processed data. Most of the sheep in the training dataset are white, but there are some brown, gray and black sheep as well. Some of the sheep are partially hidden and harder to see due to trees, vegetation and other obstacles in the grazing areas. The ground where the sheep graze is different in various parts of Norway, especially considering the solutions might be applied from April, through the summer, and as late as October. Because of this, some images might contain snow while others might be taken in tall grass after a long summer. As this thesis will discuss the usage of IR imagery to assist in this process, it is also relevant to note that the ground temperature varies greatly throughout the season. The sheep's woolliness also increases throughout the grazing season, decreasing the amount of infrared radiation the sheep are emitting compared to the ground. All of these factors show how important generality is for the use case.

4.5 Hardware

During the project, some hardware limitations have been taken into account. All images were captured with the same drone, and all training and computationally heavy operations were conducted on NTNU's HPC cluster IDUN, and additionally measured and estimated on mobile devices.

4.5.1 UAV

During data collection, a DJI Mavic 2 Enterprise Dual UAV was used to capture all the images. Detailed specs are found in Table 4.1, and an image of the drone is seen in Figure 4.1.

Table 4.1: Specs for DJI Mavic 2 Enterprise Dual edition [49].

Take-off weight	899 g (without accessories)
Max take-off weight	1100 g
Max tilt angle	35° in Sport-mode and 25° in Positioning-mode
Dimensions (unfolded)	322 mm × 242 mm × 84 mm
Max flight time	31 min (at 25 kph)
Max speed	72 kph
Thermal camera resolution	160 px × 120 px
Thermal camera HFOV	57°
Visual camera resolution	4056 px × 3040 px
Visual camera HFOV	85°

Figure 4.1: DJI Mavic 2 Enterprise Dual. Photo from DJI's website [49].

The Mavic 2 is a four propeller helicopter drone which carries a dual camera with both RGB and IR capabilities, although the IR camera only captures images at a resolution of 160 × 120 pixels, and then upscales them to 640 × 480 pixels. This low base resolution means that there is not very much infrared data, and using a camera with higher IR resolution might provide better results for the neural networks that use IR data. This is further discussed in subsection 7.2.2. The cameras are also designed to capture images while the drone stays still which limits mobility, reduces the distance covered by one battery charge and increases time between image capture.

4.5.2 IDUN

One of NTNU's high performance computing (HPC) clusters, IDUN [50], was utilized during training of the models. Through IDUN, one is given access to high end GPUs (NVIDIA P100, V100 and A100), and other high end industry hardware, which drastically lowers training- and validation times. This gives an indication of how fast a system would process images in the cloud. IDUN is a shared resource, and is managed by the Slurm Workload Manager, allowing users to queue their jobs and be notified at their completion.

The usage of IDUN does not necessarily affect the results in any noteworthy way, but using high end GPUs decreases computational time and therefore also inference time. This also means that any measurements of inference time must be taken on identical hardware to be comparable. On the other hand, these high end GPUs also allow for larger batch sizes during training due to larger memory capacity, which affects how often back propagation is run. This might affect the training results to some degree, and generally increases the training speed [51].

4.5.3 Mobile devices

In order to test how the system would perform when processing images on mobile devices, two options are considered, namely laptop computer and mobile phone. One real-world hardware device is chosen for each option to ensure repeatability. The first option is to use a laptop computer, ASUS FX502VM, with an Intel i5-7300HQ @ 2.5 GHz and 8 GB of RAM. The laptop has a dedicated NVIDIA GTX 1060 Mobile GPU with 3 GB of memory and a 128 GB M.2 SSD. This laptop is from 2017 and is therefore in the lower end of what would be available for laptops these days. The second option is to use a mobile phone. For this comparison, the Huawei P30 Pro from 2019 is used. It has a Kirin 980 Octa-core Processor with 6 GB of RAM, as well as an ARM Mali-G76 MP10 GPU. This thesis does not measure any processing times on the mobile phone, but rather estimates these results based on the findings of Furseth and Granås [15].

4.6 Software

During the work on the thesis various COTS (commercial off-the-shelf) products have been used, including Python, CUDA, Roboflow, Pytorch and YOLOv5.

4.6.1 Roboflow

Roboflow is a web application which allows a group to cooperate in annotating images using bounding boxes. It is easy to use and supports the annotation of multiple classes of objects. It is possible to upload images, and also corresponding annotations if any of the images are already annotated. Roboflow support a wide range of annotations formats, both for uploading and downloading the dataset.

4.6.2 Software environment

For development of the YOLOv5 architecture, Python version 3.9.2 is used. Python 3.8.6 is used when running jobs with training and validation on IDUN, and CUDA 11.1 is needed for running on Nvidia A100 GPUs. The YOLOv5 repository builds upon PyTorch, which is an open source machine learning framework for Python, widely used in research communities. It is developed by Facebook AI Research (FAIR). When training on IDUN, PyTorch version 1.9.0+cu111 is used, i.e. PyTorch version 1.9.0 with CUDA 11.1 support.

4.6.3 YOLOv5 repository

The YOLOv5 repository features a customizable framework for training and validation of models, aswell as detection on unseen images. The default model is the YOLOv5 architecture, but this can easily be changed through a `.yaml` config file. The framework makes it easy to use pretrained models when continuing training, and features a rich augmentation library. It also offers useful summaries, graphs and images after running training and validation runs [27].

Chapter 5

Method

Chapter five describes the approach used to answer the Research Questions and reach the goal presented in section 1.3. The initial literature review described in chapter 2 ensures that this thesis' work is conducted at the cutting edge. Based upon this existing research, an appropriate, unmodified machine learning architecture is selected, which serves as the control group for the achieved results. 12 variations of this architecture are tested and for each variation relevant metrics are recorded as results. In order to show statistical significance for the results, a variance probing experiment is conducted, which shows the distribution of results from identical runs of the base model. The final section in this chapter concerns itself with time, both with regards to the estimated time available between the capture of images, as well as the estimated time needed to process each image. All-in-all the chapter describes processes required to repeat the conducted experiments, as well as motivations for the choices made.

5.1 Dataset

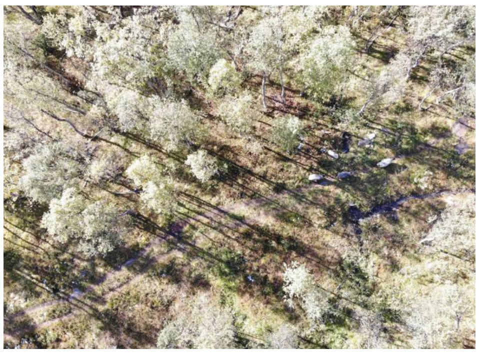
The dataset used when training and testing the models are drone image pairs of sheep, where each pair consists of one RGB image and one IR image. In Johannessen's master thesis [14], approximately 1000 images were annotated with the sheep's locations and colors. These images were captured in Storlidalen in the autumn of 2019, and in Klæbu and Orkanger in the spring of 2020. During the autumn of 2021, a co-student named Hallvard Stemshaug captured around 1000 images in an area called Holtan. In a joint effort, these images were annotated using Roboflow, the tool presented in subsection 4.6.1.

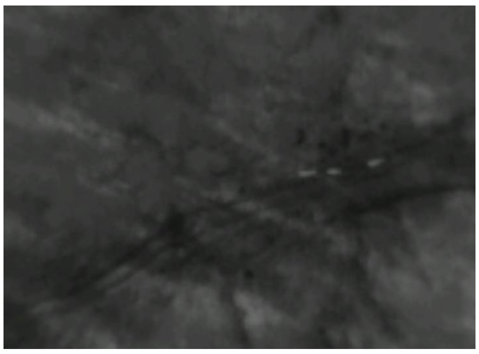
After combining the two datasets and removing blurry images, images without sheep, too similar images, and image pairs where the IR image is either damaged or in colors, the dataset consists of 2022 image pairs. Figure 5.1 shows examples of image pairs that are removed because of the IR images is somehow not preferred to train on. When capturing IR images there is an option to use a mode called MSX. When MSX is enabled, edges detected in the RGB images are overlayed on top of the IR images. In the first dataset, 431 images captured with this mode on. Because this data modification is unnecessary when processing both images, and as it might make it more complex for the model to learn how to process the IR image, these images are not used. Therefore the final dataset consists of 1591 image pairs. Table 5.1 shows an overview of the locations where the images were captured, and the number of images captured at each location.

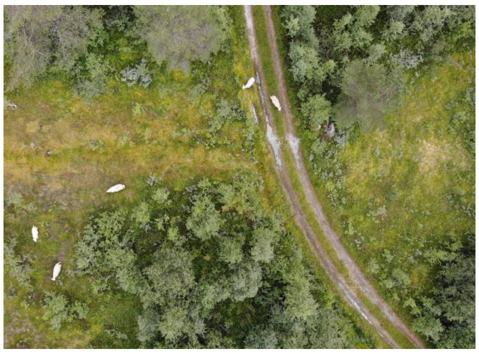
The final dataset still contains a moderate part of images with sheep grazing in a fenced area, but also contains images of sheep in rough terrain, in forest, in swamp, and in different seasons of the year and weather conditions. Figure 5.2 showcases images in different terrains. All images are taken during the day, but with varying degree of direct sunlight and shadows. The images are taken by the UAV at varying altitudes ranging from 10 to about 100 meters, though most of the images are taken at altitudes between 40 and 80 meters. There are also variations in sheep coloring and how visible the sheep are in the terrain, as they may be hidden behind vegetation. Examples of this are shown in Figure 5.3 and Figure 5.4.

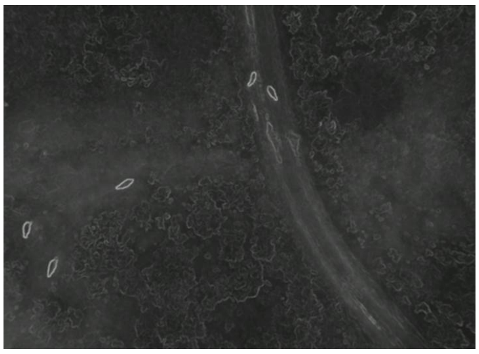
Table 5.1: Distribution of images and sheep based on capture location.

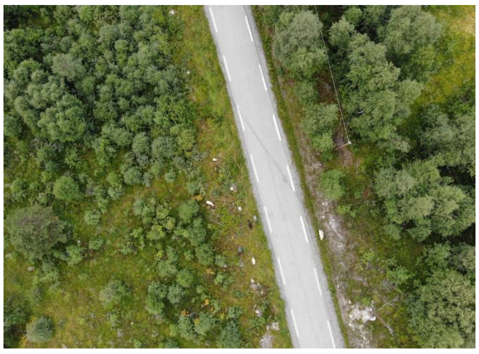
Set	**Images**	**No-MSX**	**MSX**	**Total sheep**	**White sheep**	**Gray sheep**	**Black sheep**	**Brown sheep**
Storlidalen (2019)	773	342	431	5550	3096	1809	543	102
Klæbu (2020)	106	106	0	1479	1427	2	28	22
Orkanger (2020)	98	98	0	1240	196	167	627	250
Holtan (2021)	1045	1045	0	11967	8423	1667	1188	689
Total	2022	1591	431	20236	13142	3645	2386	1063

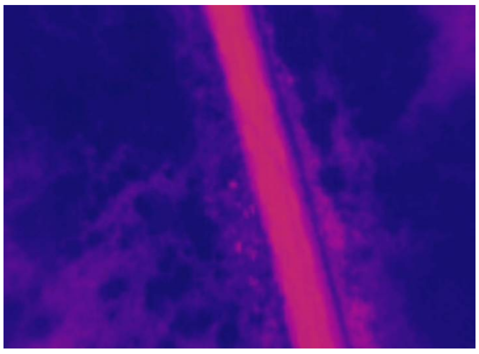

(a) 2019_09_storli2_2311 RGB

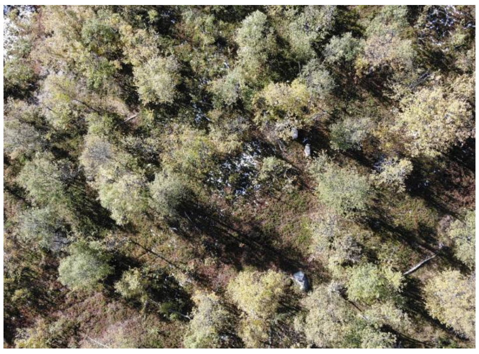

(b) 2019_08_storli2_2311 IR


(c) 2019_08_storli1_1481 RGB


(d) 2019_08_storli1_1481 IR


(e) 2019_09_storli1_1047 RGB


(f) 2019_08_storli1_1047 IR


(g) 2019_10_storli2_2155 RGB

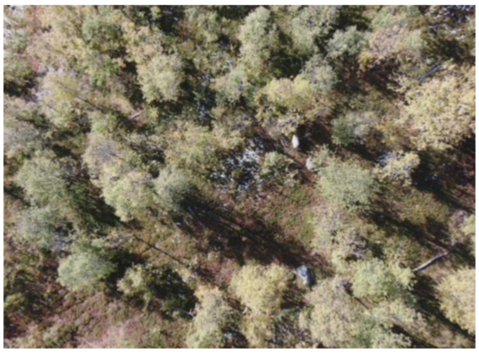

(h) 2019_08_storli2_2155 IR

Figure 5.1: Examples of RGB-IR pairs. Figure 5.1a and Figure 5.1b shows a standard RGB and IR image pair, which is accepted into the dataset. Figure 5.1d shows an example of an MSX image, which helps humans recognize structure in otherwise blurry IR images. Figure 5.1f shows a broken, wrongly colored IR image. Figure 5.1h shows a broken IR-image with RGB contents.

(a) 2019_08_storli1_0702

(b) 2019_09_storli2_2319

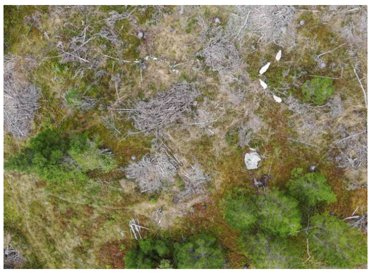
(c) 2019_10_storli3_3594

(d) 2020_05_klabo_0442

(e) 2020_05_klabo_0616

(f) 2020_05_orkanger_0854

(g) 2021_09_holtan_0761


(h) 2021_09_holtan_0865

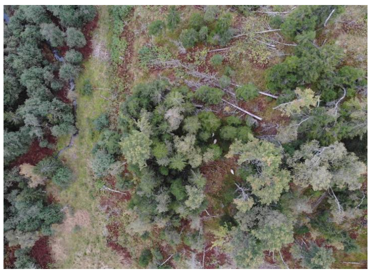
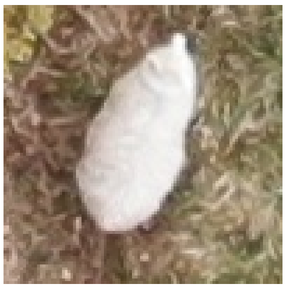
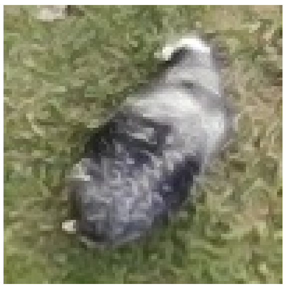
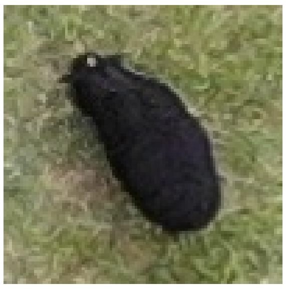

(i) 2021_09_holtan_1698

Figure 5.2: Examples of RGB images from the dataset showing some of the variations between the captured locations. The terrain varies from dense, lush forests to bare, brown fields. The different images also feature various degrees of occlusion, snow, and shadows from the sunlight and obstacles.


(a) A white sheep


(b) A gray sheep


(c) A black sheep

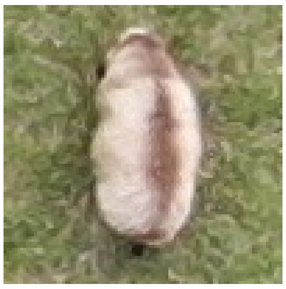

(d) A brown sheep

Figure 5.3: The four labeled colors of sheep in the dataset.

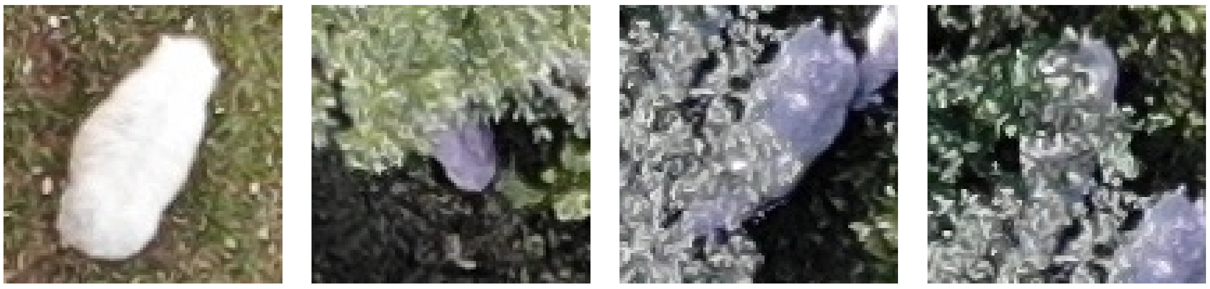


Figure 5.4: The four labeled obscurity levels of the sheep in the dataset.

5.1.1 Dataset splitting

When dividing the total dataset into training-, validation- and testing datasets, the split percent depends on the amount of images in total and the diversity of the images. A general rule for supervised learning is to divide the total dataset into 70% training data, 20% validation data and 10% test data [52]. However, the dataset is split in a less typical fashion in order to compare findings with previous results.

In total, there are 1591 usable image pairs. First, all 104 images from Klæbu (May, 2020) are separated from the rest. This subset works as a specific test dataset to compare model performance directly with Johannessen's results. The Klæbu test set also features a completely different terrain, which tests how well the model generalizes to unseen terrains. Then, from the remaining 1487 images, 125 images are selected as the validation dataset, and 125 images are selected as the ordinary test dataset including two from the Klæbu dataset. The finally remaining 1239 images are used as the training dataset. An overview of the different datasets is shown in Table 5.2.

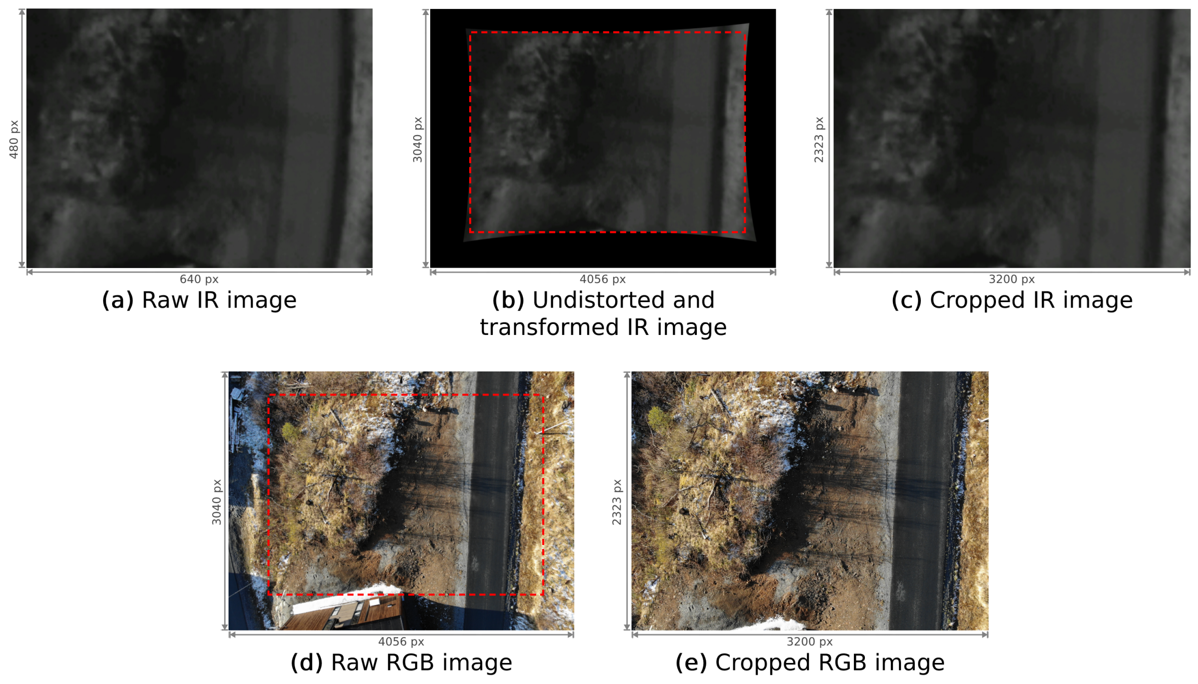
The images are often taken in series, resulting in similar images in succession, which makes it undesirable to split the images randomly as the results would be biased. Therefore the split is done manually, in order to keep images from the same series in the same dataset. It is still important to represent most of the terrain types in both the validation and test sets as well. In order to answer Research Question 3, an effort is made to include images of sheep partially hidden by vegetation in the test set.

Table 5.2: Distribution of images and sheep in the different datasets used by the model. Note that two images from Klæbu are also part of the test set, which is the reason for the 2 image discrepancy between the sum of the four datasets and the "all"-category.

Dataset	Images	Total sheep	White sheep	Gray sheep	Black sheep	Brown sheep
Train	1239	12370	7780	2138	1645	807
Validation	125	1171	907	121	90	53
Test	125	984	690	181	54	59
Klæbu	104	1216	1170	1	25	20
All	1591	15703	10509	2441	1814	939

5.1.2 Preprocessing

A model solely based on RGB images does not need any additional preprocessing. However, because a comparison between RGB, fusion and 4 channel are to be done, some preprocessing is still performed. This is done to ensure that the models train on the same data. The reason this preprocessing step is necessary, is that the RGB image and the IR image are taken with two different cameras. This results in the IR image only covering a portion of the RGB image, while also having some lens distortion. The undistortion is done with the help of coefficients and code snippet that Johannessen developed in her thesis [14]. The undistortion warps the IR image so that it follows the same lens distortion as the RGB image. Because the distortion makes the IR image non-rectangular, only a rectangular crop of the IR image is used. The RGB image is 4056 × 3040 pixels while the IR image is 640 × 480 pixels, automatically upscaled from 160 × 120 pixels on the drone. Thus, an affine transformation is used in order to scale and pad the IR image in order to line up with the RGB image. Then the RGB image is cropped to the size of the image which is also represented in the IR image. The result is an image pair of size 3200 × 2323 pixels. The process is visualized in Figure 5.5.



(a) Raw IR image **(b)** Undistorted and transformed IR image **(c)** Cropped IR image

(d) Raw RGB image **(e)** Cropped RGB image

Figure 5.5: Steps in the image preprocessing process. Subfigure **(a)** shows the raw IR image as it comes from the camera. **(b)** shows the result of the undistortion, affine transform and resizing, with the crop area highlighted. Subfigure **(c)** shows the final cropped IR image used in training. Subfigure **(d)** shows the raw RGB image, with the crop area highlighted. Subfigure **(e)** shows the final cropped RGB image used in training.

5.2 Model selection

A requirement when choosing an appropriate model is that it should be state-of-the-art, meaning it shows good results on object detection competitions, such as MS COCO [39]. Secondly, it should also be fast enough to potentially be able to run on mobile hardware in order to save network traffic. The model should also have an active community using it, to make it easier to find answer to questions while modifying it. Finally the model should be easy to put into production.

After looking into different models, YOLOv5 was chosen as the most appropriate architecture [26]. Models with this architecture show good results on MS COCO, and has a very low inference time. It also comes in a flexible framework [27], which allows developers to easily configure the model based on their needs. The framework makes it easy to train, detect and validate the models, and the training process features a rich collection of augmentations, which makes the model better at detecting sheep in different terrain and different conditions.

5.2.1 Model size

In the YOLOv5 framework it is possible to choose model configuration of different sizes. A larger model will give higher accuracy, while also increasing the inference time, whereas a smaller model will sacrifice accuracy to decrease inference times. Table 5.3 shows a benchmark of the different model sizes.

Table 5.3: An overview of the YOLOv5 model sizes, trained and validated on Microsoft COCO [39]. Inference time is measured using an NVIDIA V100. Source: YOLOv5 repository [27]

Model	**Size** (px)	**mAP** [0.5:0.95]	**mAP** [0.5]	**Inference** (ms)	**Parameters** (M)
YOLOv5s	640	37.2	56.0	6.4	7.2
YOLOv5m	640	45.2	63.9	8.2	21.2
YOLOv5l	640	48.8	67.2	10.1	46.5
YOLOv5x	640	50.7	68.9	12.1	86.7
YOLOv5s6	1280	44.5	63.0	8.2	16.8
YOLOv5m6	1280	51.0	69.0	11.1	35.7
YOLOv5l6	1280	53.6	71.6	15.8	76.8
YOLOv5x6	1280	54.7	72.4	26.2	140.7

Depending on the use case, one has to consider the trade-off between improved results and improved inference times. In one of the proposed solutions, where inference will be run on an external networked server with better hardware, the larger models are favorable. In the other proposed solution, the smaller models are favorable, as they allow the network to keep up with the incoming stream of images. The smaller models are necessary, because the inference will be run on mobile devices, on which inference times tend to be much higher [15]. For these two use cases, the two models YOLOv5l6 and YOLOv5s are selected for further experimentation. This choice of model sizes will also help answer Research Question 1 and Research Question 2.

5.2.2 Input image size

When determining the size of input images, it is important to balance two aspects. Firstly, as the sheep are relatively small in the UAV images, it is preferred to give the network as large images as possible as input, to be able to detect the sheep. A sheep has a typical bounding box of 50×25 pixels in the original UAV image, which is 4056×3040 pixels. Secondly, it is helpful to scale the images down to allow the GPU memory to contain the necessary image data for larger batch sizes during training and inference. The YOLOv5 repository has pre-trained models for image sizes 640 and 1280 pixels, as seen in the Table 5.3. Results from Johannesen's thesis [14] imply that greater image sizes generally give better results. Therefore the input image size is selected to be 1280 pixels, serving as a compromise between improved results and improved inference times.

5.3 Model improvements

In order to answer the research questions posed in section 1.3, some modifications have to be made to the image processing pipeline. The model's fitness function is adjusted, and the model is changed in order to also process IR images. In addition to model changes, one proposed improvement is implemented as both a pre- and postprocessing step, namely tiling.

5.3.1 IR images

In order to achieve the best possible result, it might be beneficial to utilize both the RGB and IR image captured by the drone. Testing this will help answer Research Question 1 and Research Question 2. Infrared images (IR) capture infrared radiation which is not visible to the human eye. The warmer a specific surface is, the more infrared radiation the surface radiates, and the camera is able to detect this. The infrared images are represented with dark colors where the surface is cold, and lighter colors where the surface is relatively warmer. As long as the sheep is warmer than the ground, the sheep will be visible as brighter silhouettes in the IR image. It can therefore be beneficial to use this extra source of information, especially when the color of the ground is almost the same as the sheep in the RGB image. Additionally, this extra information might strengthen the confidence for an actual sheep detection and diminish the confidence if it is only a rock.

In order to implement fusion in the neural network two different methods have been proposed:

1. Load both the RGB image and the IR image, and run two different pipelines for the images in the backbone, before fusing the results and run the neck and head on the fused results. This also includes fusing potential shortcuts from the backbone to the neck of the network.
2. Load both the RGB image and the IR image, but before running inference, concatenate the 3 channels from the RGB image with 1 channel from the IR image into a 4-channel image. Then change the first layer of the model to accept 4 input channels.

5.3.1.1 Fusion

In order to allow the YOLOv5 repository to train models with the use of IR images some modifications have to be made. Firstly the dataloader has to be modified to also load IR images. This also includes caching the IR images, which effectively speeds up the training, as it prevents unnecessary reading from the disk. Augmentations were also modified to properly augment the IR images, while still keeping image pairs aligned. The network architecture needs to be prepared for using fusion as described in section 3.7. In order to have one backbone for the RGB images and one backbone for the IR images, a deep copy of the original backbone is performed. This is needed for the IR backbone to have its own weights and biases, which will be changed during training. The inference function of the model is modified to take in two parameters instead of one. Then the RGB and IR data is sent through the respective backbones. Inspired by Johannessen's thesis [14], results from each backbone are then fed through a fusion layer, as seen in Figure 5.6 and previously explained in section 3.7. The fusion layer consists of a concatenation of the results from the RGB and IR backbones and a 1×1 convolution which dimension reduces the results to the original size. This makes the input to the neck a combination of the results, while still having the same dimensions, allowing it to be processed as in the original architecture.

As the YOLOv5 architecture also has some CSP skip connections from the backbone into the neck, the results going through these skip connection also has to be processed through fusion layers. This can be seen in Figure 5.6 which shows the modifications made to the original network structure, previously shown in Figure 3.3.

5.3.1.2 4 channel

The 4 channel network is a more naive way of utilizing the extra information from the infrared images. It uses the same modifications for loading the IR images as the fusion model, the difference is in the network architecture. Instead of having two backbones, the 4 channel network allows the first layer to take a 4 channel tensor as input. In the inference function, the 3 channels from the RGB image is concatenated with 1 channel from the grayscale infrared image. For the rest of the inference the network is run as the original YOLOv5 architecture.

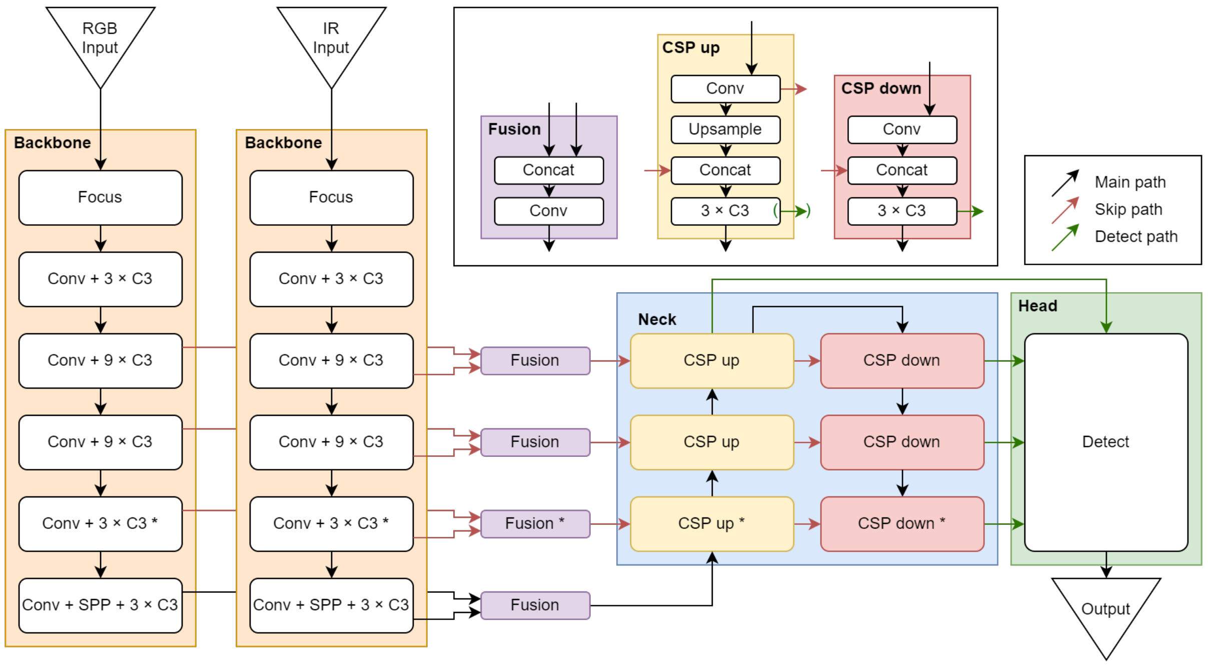


Figure 5.6: A modified version of Figure 3.3, including the fusion layers and both RGB and IR backbones. Boxes marked with an asterisk (*) are not present in the YOLOv5s architecture, but only present in the YOLOv5l6 architecture.

5.3.2 Fitness function

The fitness function serves the purpose of calculating the model's performance, as mentioned in section 3.1. Its output is used to decide whether the training should continue or halt, depending on the model's patience parameter. The default combination of metrics for the fitness function when training a YOLOv5 model, is based on the weighted average of mAP0.5 and mAP0.5:0.95, weighted 10% and 90% respectively. This setting prioritizes getting a perfect bounding box around the object detected, and hardly rewards correct bounding boxes if they are not perfectly placed. As the thesis' use case does not require the predicted bounding boxes to overlap perfectly with the ground truth, this weighting is changed to rely 10% on the recall metric, and 90% on mAP0.5. This change encourages the network to achieve a higher recall and mAP0.5, of which the latter is a good metric for balancing confidences in order to maximize both precision and recall. This is further explained in subsection 3.8.3. This change does come at the cost of the network no longer working to improve the mAP0.5:0.95, which is found to be an acceptable compromise.

5.3.3 Tiling

As presented in section 3.6, it might be easier for the neural network to detect sheep if they are larger. It can therefore be relevant to analyze smaller tiles of the full image, in order to avoid downscaling before running the network. Examining and testing this will help in answering Research Question 1 and Research Question 2.

When utilizing tiling, both the images and the corresponding labels must be tiled before training. In order to compare model performance with non-tiled models, the predictions made during inference must be combined to correspond to a full image.

5.3.3.1 Image tiling

The image tiling process is performed in three steps, as seen in Figure 5.7:

1. Load the original image.
2. Find coordinates for the 6 sub-images with overlap areas.
3. Divide the original image into subimages.

The preprocessed RGB- and IR images are 3200×2323 pixels large, and are each divided into six subimages of 1200×1200 pixels which are then run in a model trained on images of this size. By both training and running inference on these smaller tiles, the model will hopefully have an easier job detecting sheep. The final size of the images are seen in Figure 5.7, and the overlap is apparent in the result.

5.3.3.2 Label tiling

The labels are tiled in the same fashion as the images, following similar strategies. Labels which fall into multiple tiles are all kept, meaning some labels might not be correct in all the tiles. An example can be seen in Figure 5.7, where sheep 3 is divided into tile T00, T01, T10 and T11. The label in T10 is kept as normal, while labels in T00 and T11 might be a bit difficult for the network to work out - but not impossible. The label in T01 will, however, be quite impossible for the network to learn, as there are no pixels from the sheep present in the photo. This might lead to some issues when training, given that the process will attempt to teach the model to recognize a sheep there, without any of the sheep landmarks being present within the bounding box. It was decided, considering that this happens fairly infrequently, to leave these potentially wrongful labels in the training dataset. This is further discussed in subsection 7.2.1.

5.3.3.3 Prediction combining

When inferencing and running detection on tiled images, the detections need to be combined, so they reference the coordinate system of the original image. The network might find the same sheep present within the overlapping areas of multiple images. Using sheep 2 in Figure 5.7 as an example, the network might notice the sheep in tiles T00 and T10. These labels are both correct, but having two will make one of them count as a false positive. To remedy this, the non-maximum suppression algorithm explained in section 3.9 is applied. As presented before, the NMS IoU threshold used is 0.45, meaning for label pairs with an IoU > 0.45, the least confident label is removed. For sheep 2, this will probably work out nicely for a well behaving network, and only the most confident label for sheep 2 will be kept.

An issue with this algorithm, for this use case, is that it will not always work as desired. E.g. in Figure 5.7, sheep 5 is visibly divided between T11 and T21. The detectable portion found in T11 is approximately 25% of the portion found in T21, meaning the detected IoU will be around 0.25, which is less than the threshold of 0.45. In this example neither label is removed, and the one with the lowest IoU with the ground truth is considered a false positive.

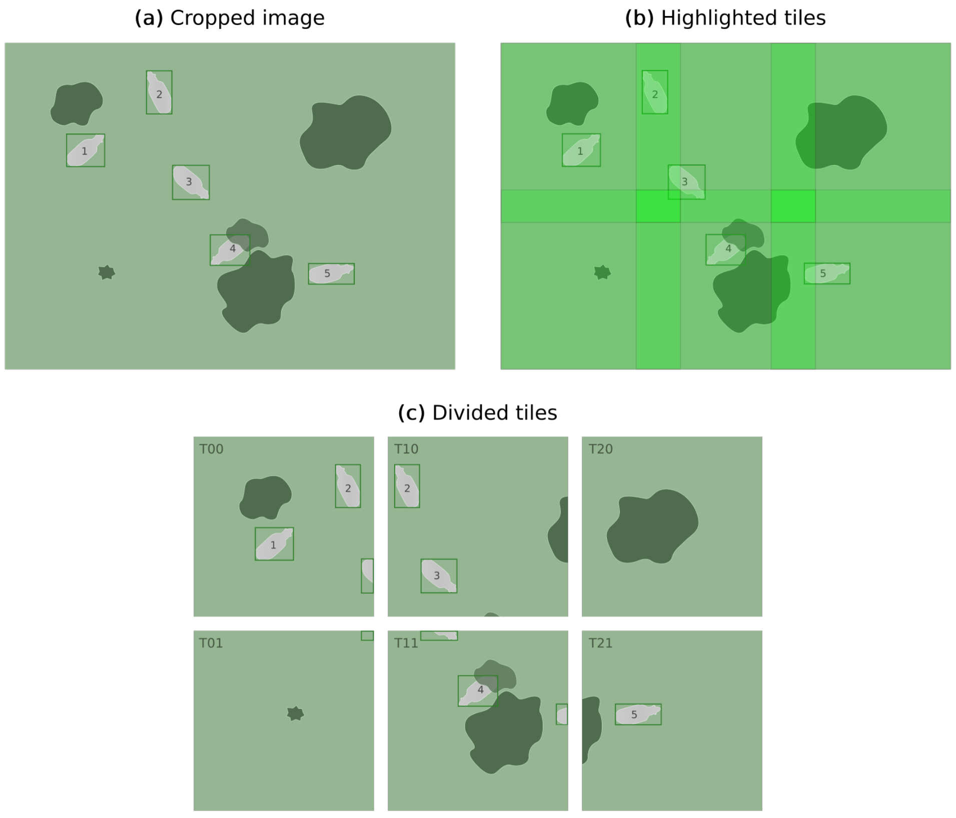


Figure 5.7: The three stages of the tiling process. Green bounding boxes represent ground truth labels. Tile names are formatted as T[X][Y].

5.4 Main experiment

To answer the Research Questions presented in section 1.3, models have to be trained. Given the three color modes RGB, RGB + IR fusion and RGB + IR as four channel input, two tiling modes (with and without tiling) and the two model sizes YOLOv5s and YOLOv5l6, there are $3 \times 2 \times 2 = 12$ necessary experimental configurations.

All models are trained with the static settings as seen in Table 5.4. They are trained from scratch without any pretrained weights and with default hyperparameters from the YOLOv5 repository. The models are exposed to images from the training set of size 1280 pixels, with a batch size of 8. For every epoch an evaluation on the validation set is performed. A patience of 100 means that the training will stop early if the results have not improved during the last 100 epochs. The maximum number of training epochs is set to 2000. Training is performed on IDUN nodes with a NVIDIA A100m40 or NVIDIA A100m80.

Table 5.4: Table displaying the identical settings used for each training run.

Property	Value
Image size	1280px
Batch size	8
Max epochs	2000
Patience	100
GPU	NVIDIA A100

5.5 Variance probing

As YOLOv5's augmentation and weight initialization is based on a random number generator, the results for similar runs are rarely the same. The concept of augmentation is presented in section 3.5. To probe how different the results tend to be, 50 identical training runs are conducted on the same dataset, using the same hyper parameters and settings. Each run uses the YOLOv5l6 model size, using only untiled RGB data as input. The settings used for the variance probing are the same as the ones used in the main experiment, as seen in Table 5.4. These experiments are run in order to evaluate how certainly one can describe a model improvement, based on empirical study. Our method is based on the work done by Bouthillier et. al [53] on the probability of outperforming. The null hypothesis, as described in section 3.10, for this study is that each modification has no effect on the results.

Assuming that a stochastic value follows a normal distribution, one can calculate the probabilities of a single datapoint falling within n standard deviations of the distribution's mean. This is further explained in section 3.10. By assuming that the mAP0.5 follows a normal distribution, the probability of one experimental result actually showing improvement over another can now be described, as opposed to it simply being a stochastic coincidence. Whether or not it is reasonable to assume these metrics follow the normal distribution will be discussed in section 7.1. For this thesis, a statistical significance level of $\alpha = 4.6\%$ is selected, corresponding to the normal distribution's $\mu \pm 2\sigma$ limit values.

5.6 Evaluation

In order to evaluate the performance of the models they must be evaluated on a test set and compared using a set of requirements and metrics, as described in section 4.4. In subsection 5.3.2, mAP0.5 is chosen to be the main focus of the fitness function, as it is deemed more appropriate considering the localization requirements of the task. As it is often more intuitive to understand precision and recall, they are also included in the evaluation.

In addition, to compare results against Johannessen, a script to transform the bounding box predictions into grid based prediction is developed. This process is described in subsection 3.8.4. Therefore the evaluation also contains measurements of average precision, precision and recall for the grids. The average precision for grids is measured using the function `average_precision_score()` within the `scikit-learn.metrics` package, which Johannessen describes in her thesis [14]. In addition, grid sheep recall is measured, which is based on the number of sheep in the detected grids out of the total of all sheep.

To answer Research Question 3, the recall for the different sheep colors and for the different types of sheep hidden by vegetation are also measured. Each sheep bounding box is categorized according to the sheep's color, and the colors used are white, gray, black and brown. The different categories for sheep hidden by vegetation are partially obscured, completely obscured, partially covered and in the open.

5.7 Time estimation

To answer Research Question 2 regarding the time constraints, it it necessary to have a more concrete measure for the time requirement in section 4.4. Therefore, an estimation of the time spent moving a drone between image capture locations is performed. These results are then compared to numbers describing the time needed for different models to run on different hardware, to see what types of solutions are viable for a system application.

5.7.1 Time between shots

The first task is to calculate the distance an image covers on the ground, and thus the distance the drone has to travel before capturing the next set of images. From the specifications for the drone [49], the IR camera's horizontal field of view (HFOV) angle is 57°, which is less than the RGB camera's HFOV angle at 85°. As the shortest distance between captured locations will be along the vertical axis of the images, the vertical field of view (VFOV) is more relevant, and can be calculated from the HFOV by solving the equation set given in Equation 5.1.

$$\begin{aligned} k \times tan(\frac{57^\circ}{2}) &= \frac{160px}{2} \\ k \times tan(\frac{\text{VFOV}}{2}) &= \frac{120px}{2} \end{aligned} \tag{5.1}$$

By solving Equation 5.1, it is found that $\text{VFOV}_{\text{IR}} = 44.3^\circ$, and assuming a height of 80m, it can be calculated that the distance covered by the vertical field of view is 65 meters. See section 3.11 for details into this calculation.

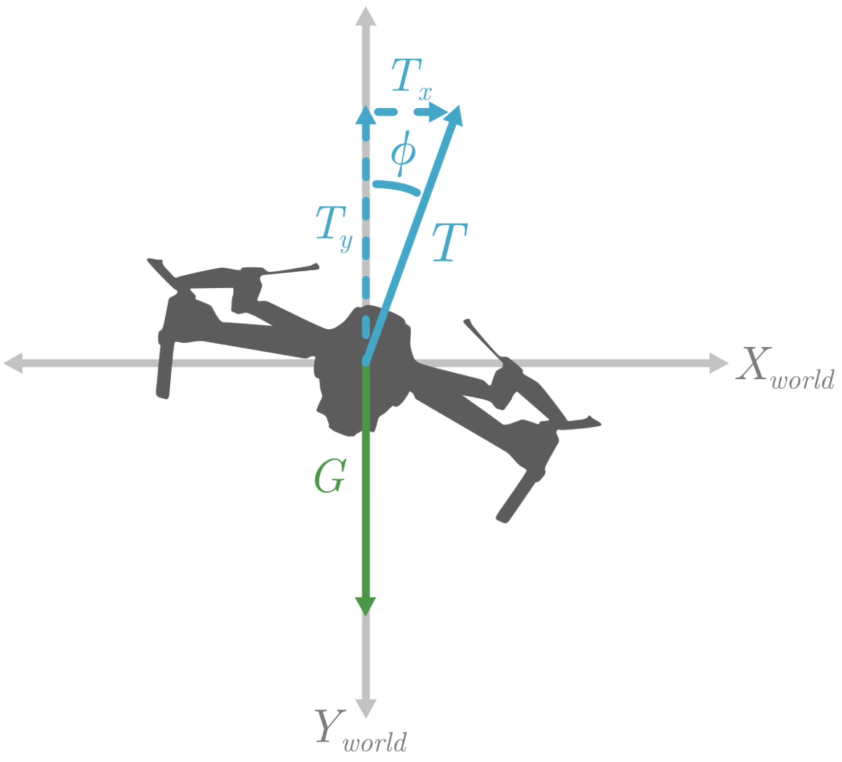


Figure 5.8: The forces affecting a drone while tilted.

Given that a helicopter drone needs to make a full stop during photo capture, the average speed will not simply be the top speed of the drone. In order to find an estimation of the time interval between image captures, the acceleration of the drone has to be found. The horizontal acceleration of a drone (a_y) is a combination of the amount of thrust from the motors (T) and the tilt of the drone (ϕ), as seen in Figure 5.8. The Mavic 2 Enterprise Dual has a maximum tilt angle of 25° for the positioning mode. In a production environment the positioning mode will probably be the activated mode, as it enables GPS positioning and vision systems. In order to do an overestimation, however, the maximum tilt angle for sport mode will be used, which is 35°. An estimation for maximum thrust can be found by assuming that the drone can barely produce enough vertical thrust when using maximum tilt with maximum weight of 1100g. By applying the formula in Equation 5.2, we find that the maximum thrust is 13.2N.

$$\begin{aligned} T_y &= G \\ T \times cos(\phi) &= mg \\ T &= \frac{mg}{cos(\phi)} \end{aligned} \tag{5.2}$$

Then the acceleration of $8.4\frac{m}{s^2}$ can be found by using the formula in Equation 5.3 with normal weight of $899g$ and maximum tilt of 35°. By assuming a constant acceleration and deceleration, the drone uses about $2.4s$ to reach top speed of $20\frac{m}{s}$, and roughly the same when halting.

$$\begin{aligned} T_x &= T \times sin(\phi) \\ ma_x &= T \times sin(\phi) \\ a_x &= \frac{T \times sin(\phi)}{m} \end{aligned} \tag{5.3}$$

For the drone to cover the 65m of the vertical image, an estimation of velocity relative to the distance covered is visualized in Figure 5.9. The time it takes to accelerate and decelerate the drone is $2.4s$, and the drone spends $(41m-24m)/20\frac{m}{s} = 0.85s$ at top speed. Thus the total time spent on moving in between capturing images is $2.4s + 0.85s + 2.4s = 5.65s$. During optimal operations, with no unnecessary waiting, this means the preprocessing, inference and postprocessing of the results will have less than $5.65s$ seconds to complete.

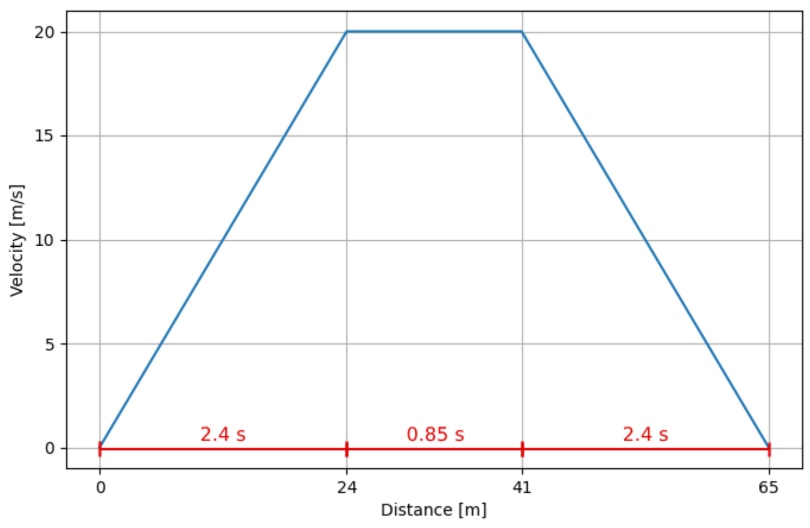


Figure 5.9: A Mavic 2 Enterprise's velocity on its path between two image capture locations. The x-axis shows the distance traveled, and in red the time spent on each segment is shown.

5.7.2 Processing times

The total processing time of an image depends on the processing device, and having a dedicated GPU significantly reduces the time it takes to run inference on images [54]. Therefore measurements are performed on the IDUN cluster and on a laptop with dedicated GPU. Additionally, estimations are made for a specific mobile phone's performance based on results found in Furseth and Granås' thesis [15]. Table 5.5 shows an overview of the used devices' GPUs and CPUs.

The processing time is divided into preprocessing, inference and postprocessing. Preprocessing time is the time it takes undistort and crop an image, and when tiling is enabled the image is also split into 6 tiles. Inference time is the time the machine learning model spends calculating its predictions, and for tiled models it is the time spent on running inference for all 6 tiles. Postprocessing time is the time the model spends on non-maximum suppression, and for tiled models also the time spent combining tiled predictions.

In order to find an estimated ratio between the laptop and the mobile phone, measurements are done using the same model size and image sizes as in Furseth and Granås' thesis [15], measured on a laptop with a dedicated GPU. The ratios are found by dividing the mobile inference times by the corresponding laptop inference times, showing how many times slower inference on the mobile device is. The average of the ratios is then calculated, and used as a ratio to multiply the laptop inference time to get an estimation for the time on a mobile phone. In addition, an estimation for the preprocessing and postprocessing times for the mobile phone is used. As these pre-/postprocessing scripts are not optimized for GPU computing, an assumption is made that the mobile results do not differ too much from the laptop results. This assumption will be discussed in subsection 7.1.2.

Table 5.5: Specifications for the devices used for processing time measurements.

	CPU	**GPU**
IDUN	Intel Xeon Gold 6248R	NVIDIA A100 40Gb
Laptop	Intel i5-7300HQ	NVIDIA GTX 1060 Mobile
Mobile phone	Kirin 980 Octa-core Processor	ARM Mali-G76 MP10

Chapter 6

Results

After running the experiments explained in chapter 5, the next step is to present the objective findings in understandable and easily comparable ways. This chapter presents these results, and attempts to visualize them to ease the understanding of them.

6.1 Main experiment

The models from the 12 training runs are validated on both the test dataset and the Klæbu dataset, as described in section 5.4. The results from these two validations are found in Table 6.1 and Table 6.2, respectively. These results will mainly help in answering Research Question 1. The results are divided based on the different model configurations, namely the input image type, tiling and base model architecture. The best values in each column are highlighted in bold, where relevant.

The table headers for Table 6.1 and Table 6.2 are explained as follows: *Inference* shows the time, in milliseconds, spent by the neural network performing inference per image. For the models with tiling enabled, this time is the sum of running inference on all tiles, i.e the time it takes to run inference on the entire raw image pair. The time does not include preprocessing and postprocessing. Under the headline *Bounding box metrics*, lie general metrics used for most bounding box object detection networks. *mAP0.5* shows the average precision, given an IoU threshold of 0.5. *Conf. thr.* is the confidence threshold picked to optimize the F1, and this confidence is used by both bounding box *Precision* and *Recall*, as well as for all the grid metrics. Under the headline *Grid metrics* are all the metrics used to compare with Johannessen's results [14]. The *AP* is the grid average precision, in the same way that *Precision* and *Recall* here refer to the gridded versions. *Sheep recall* refers to the number of sheep detected, given that all sheep within activated grid cells are counted as found.

In Table 6.1, the highest mAP0.5 is 0.957, achieved by both the fusion and 4 channel models with YOLOv5s as the base model and with tiling enabled. The best grid average precision is 0.943 for the large tiled 4 channel model, while the best grid sheep recall at 0.975 is achieved by the small RGB model with tiling enabled. When the models are validated on the seperate Klæbu set, the models do not achieve as good results. Table 6.2 shows that the highest mAP0.5 achieved is 0.910, by the large, tiled fusion model. This fusion model also has the best grid average precision with 0.974 and in addition the best grid sheep recall with 0.988. A sheep recall of 0.988 is also achieved by the 4 channel model based on YOLOv5s when using tiling.

An easier-to-read visualization of the mAP0.5 values can be seen in Figure 6.1. These scatter charts show the corrolation between increased inference times and improved performance in the two test cases. An optimal model would score a high mAP0.5 and have a low inference time, meaning the data point would tend towards the top left of the chart. A pareto line has been drawn to indicate which models are "pareto efficient", and worth pursuing - according to the experiments. Within the YOLOv5l6 cluster, an extra point is drawn in for each dataset. This point symbolizes the average mAP0.5 from the variance probing experiment, which is presented in section 6.2. A ruler is also

Table 6.1: Results from the 12 combinatorial runs when validated on the test dataset. Shows inference times on an NVIDIA A100 40GB, bounding box metrics and grid metrics.

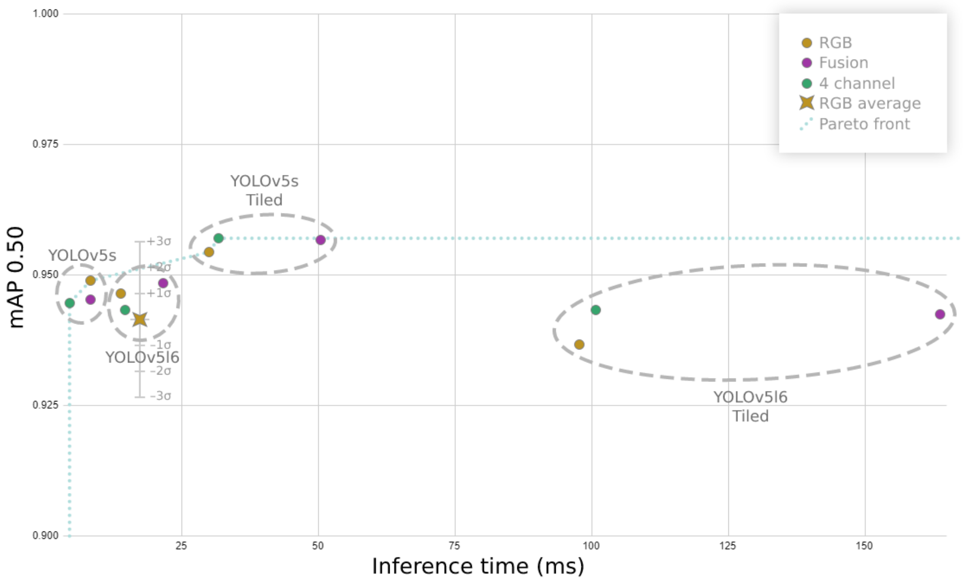
				Bounding box metrics				**Grid metrics**			
Input	**Tiled**	**Base model**	**Inference [ms]**	**mAP0.5**	**Conf. thr.**	**Precision**	**Recall**	**AP**	**Precision**	**Recall**	**Sheep recall**
RGB	False	YOLOv5s	8.3	0.949	0.639	**0.961**	0.903	0.939	0.964	0.936	0.964
	False	YOLOv5l6	13.8	0.946	0.665	0.933	0.894	0.925	0.961	0.924	0.954
	True	YOLOv5s	30.0	0.954	0.725	0.938	**0.926**	0.949	0.949	**0.954**	**0.975**
	True	YOLOv5l6	97.8	0.937	0.778	0.916	0.903	0.931	0.906	0.938	0.965
RGB + IR (fusion)	False	YOLOv5s	8.3	0.945	0.557	**0.961**	0.899	0.923	**0.974**	0.920	0.954
	False	YOLOv5l6	21.6	0.948	0.436	0.940	0.904	0.934	0.945	0.940	0.964
	True	YOLOv5s	50.4	**0.957**	0.732	0.943	0.916	0.941	0.966	0.945	0.970
	True	YOLOv5l6	163.8	0.942	0.737	0.945	0.867	0.919	0.968	0.917	0.951
RGB + IR (4 channel)	False	YOLOv5s	**4.4**	0.945	0.598	0.959	0.904	0.941	0.950	0.940	0.966
	False	YOLOv5l6	14.5	0.943	0.588	0.956	0.890	0.925	0.947	0.924	0.956
	True	YOLOv5s	31.8	**0.957**	0.728	0.945	0.915	**0.943**	0.961	0.947	0.972
	True	YOLOv5l6	100.8	0.943	0.703	0.913	0.908	0.941	0.918	0.949	0.972

Table 6.2: Results from the 12 combinatorial runs when validated on the Klæbu dataset. Shows inference times on an NVIDIA A100 40GB, bounding box metrics and grid metrics.

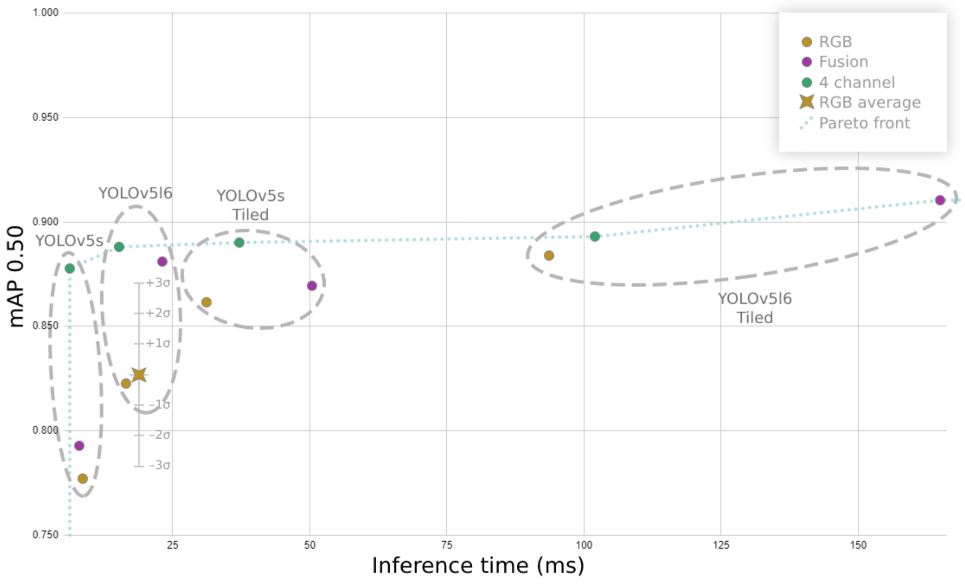
				BoundingBox metrics				**Grid metrics**			
Input	**Tiled**	**Base model**	**Inference [ms]**	**mAP0.5**	**Conf. thr.**	**Precision**	**Recall**	**AP**	**Precision**	**Recall**	**Sheep recall**
RGB	False	YOLOv5s	8.6	0.777	0.329	0.854	0.685	0.821	0.914	0.855	0.875
	False	YOLOv5l6	16.6	0.823	0.472	0.899	0.715	0.849	0.924	0.851	0.865
	True	YOLOv5s	31.2	0.862	0.660	0.881	0.820	0.910	0.893	0.958	0.971
	True	YOLOv5l6	93.6	0.884	0.732	0.866	0.856	0.947	0.886	0.972	0.983
RGB + IR (fusion)	False	YOLOv5s	7.9	0.793	0.438	0.913	0.683	0.902	0.946	0.911	0.880
	False	YOLOv5l6	23.2	0.881	0.457	0.925	0.795	0.934	0.952	0.940	0.968
	True	YOLOv5s	50.4	0.869	0.564	0.869	0.818	0.940	0.931	0.966	0.980
	True	YOLOv5l6	165.0	**0.910**	0.396	0.892	0.865	**0.974**	0.936	**0.981**	**0.988**
RGB + IR (4 channel)	False	YOLOv5s	**6.2**	0.878	0.413	0.903	0.820	0.923	0.953	0.927	0.962
	False	YOLOv5l6	15.3	0.888	0.489	**0.934**	0.809	0.925	**0.966**	0.923	0.956
	True	YOLOv5s	37.2	0.890	0.503	0.877	**0.873**	0.953	0.929	0.978	**0.988**
	True	YOLOv5l6	102.0	0.893	0.528	0.875	**0.873**	0.947	0.923	0.951	0.978

drawn in, which represents the standard deviations found in the variance experiment. This tool will be helpful in the discussion.

In Figure 6.1a, it can be seen that the inference time is longer when the model size is larger and tiling is enabled. The fastest models are the YOLOv5s-based models without tiling. It can also be seen that the YOLOv5l6 tiled models perform worse than the others even though they have the highest inference time, while the tiled YOLOv5s models achieve the highest mAP0.5 without increasing the inference time too much. Notice that in Figure 6.1b, the mAP0.5 y-axis cover a larger area than for the test set, meaning the models' performances are more spread out. Figure 6.1b shows the 4 channel model lying on the pareto front for the three groups with lowest inference times, while the model achieving the highest mAP0.5 score is the large, tiled fusion model, as previously seen in Table 6.2.



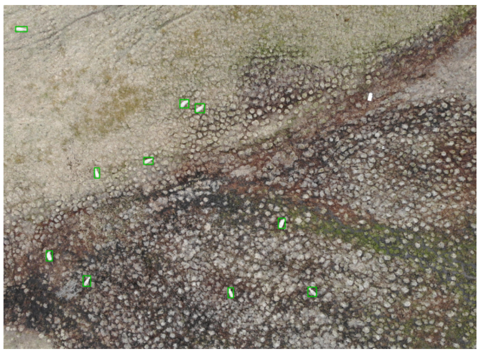
(a) Scatter plot for the test dataset.



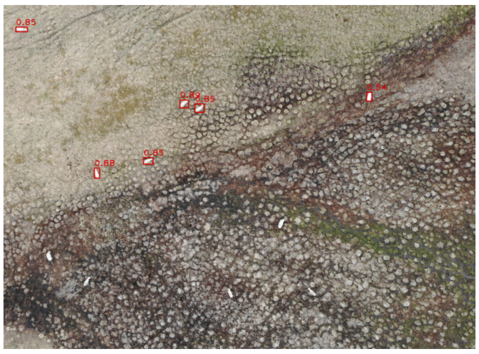
(b) Scatter plot for the Klæbu dataset.

Figure 6.1: Scatter plot of the relation between mAP0.5 values and inference times for the 12 models, on both the test and the Klæbu datasets.

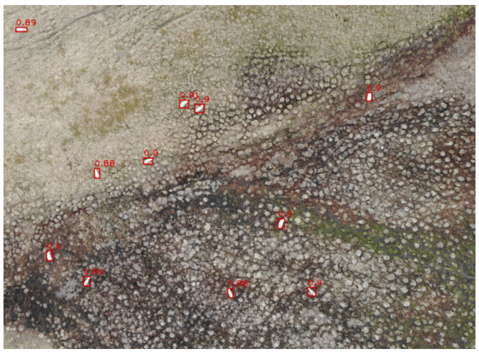
In Figure 6.2, different models' detections for the image 2020_05_klabo_0440 from the Klæbu dataset are shown. This particular image was chosen because most of the trained models struggled with detecting the sheep in the image correctly. The 10 ground truth labels of sheep are shown in Figure 6.2a, whilst the other images show detections from models with various architectures and tiled settings. In Figure 6.2b the RGB-model detects 5 sheep, while the tiled model in Figure 6.2c successfully detects all 10 sheep, but both additionally detects the sheep-like piece of debris in the top right corner. The fusion model in Figure 6.2d detects 9 of the sheep, while the tiled model in Figure 6.2e detects all the sheep including the non-sheep object, though with a lower confidence than the tiled RGB model. The 4 channel model in Figure 6.2f detects 6 of the sheep, while the tiled model in Figure 6.2g detects all 10 sheep, but also detects two false positives.



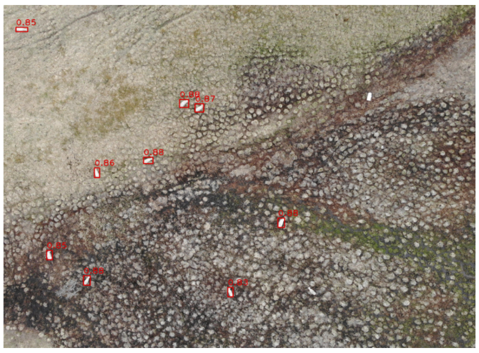
(a) Ground truth



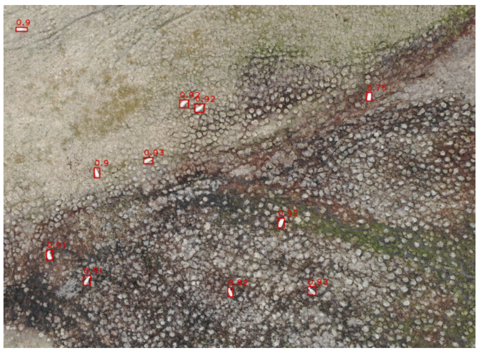
(b) RGB model



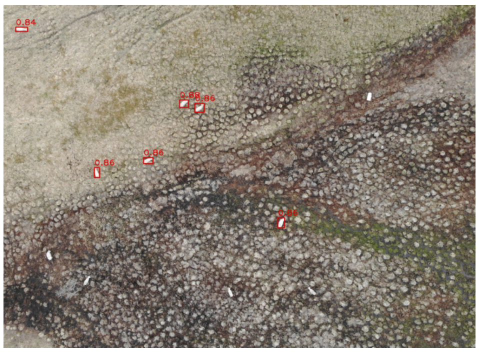
(c) Tiled RGB model



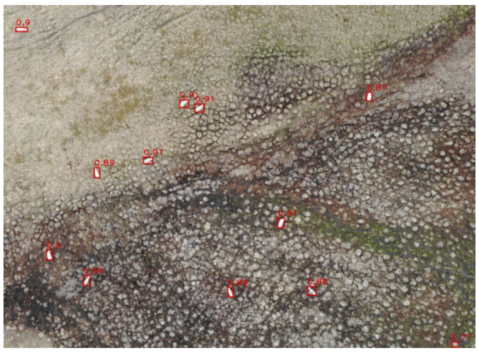
(d) Fusion model



(e) Tiled fusion model



(f) 4 channel model



(g) Tiled 4 channel model

Figure 6.2: Comparison of model detections on image 2020_05_klabo_0440 from the Klæbu dataset. All models are of the size YOLOv5l6. Green bounding boxes are ground truth, while red bounding boxes are predictions with a confidence score above. Note the differences in the prediction bounding boxes between the images.

6.2 Variance probing

The variance probing experiment is performed in order to prove that the improved results following the model adjustments are not just coincidences, as further explained in section 5.5. The results of this experiment are shown in Table 6.3 and Table 6.4, for the test and Klæbu datasets, respectively. This helps in answering Research Question 1 with a statistical significance. As mentioned in section 5.5, these results were gathered over 50 identical runs, in order to quantify the differences in the results caused by the randomized aspects of the model training. The metrics for each run in the variance probe experiment are available in Appendix A.

Each row represents a different meta-statistic for the recorded data in each column. Because of this, comparing two cells is not especially interesting, but the table serves mostly as a lookup table in order to say something meaningful about the main experimental results. In the table, *std. dev.* is the standard deviation for each metric, *average* refers to the average value. *Min* and *max* are respectively the smallest and largest values in the column.

Table 6.3 shows that the mAP0.5 for the test dataset varies from minimum 0.930 to maximum 0.955 with an average value of 0.941 and a standard deviation of 0.005. It also shows that the grid average precision varies in the range of 0.891 to 0.936 with an average of 0.912 and a standard deviation of 0.010. In Table 6.4, it is shown that for the Klæbu dataset, the mAP0.5 varies from a minimum of 0.791 to a maximum of 0.85, averaging out at 0.826, with a standard deviation of 0.015. The table also shows that the grid average precision varies in the range of 0.844 to 0.899 with the average being 0.873 and the standard deviation ending up at 0.014.

An interesting way to visualize bounding box mAP0.5 results from the variance probe, are to show their distribution using histograms, which is seen in Figure 6.3.

Table 6.3: Cumulative results from the 50 identical RGB training runs, when validated on the test dataset. The full results are listed in Appendix A.

	Bounding box metrics				**Grid metrics**			
	mAP0.5	**Conf. thr.**	**Recall**	**Precision**	**AP**	**Precision**	**Recall**	**Sheep recall**
Std. dev.	0.00520	0.05089	0.01283	0.01142	0.01018	0.01100	0.01116	0.00835
Average:	0.94111	0.64538	0.94836	0.88560	0.91163	0.95466	0.90875	0.94461
Min:	0.93038	0.51600	0.92112	0.86063	0.89129	0.93260	0.88536	0.92981
Max:	0.95545	0.75800	0.97330	0.90743	0.93558	0.97909	0.93298	0.96101

Table 6.4: Cumulative results from the 50 identical RGB training runs, when validated on the Klæbu dataset. The full results are listed in Appendix A.

	Bounding box metrics				**Grid metrics**			
	mAP0.5	**Conf. thr.**	**Recall**	**Precision**	**AP**	**Precision**	**Recall**	**Sheep recall**
Std. dev.	0.01543	0.07143	0.01660	0.02365	0.01411	0.00734	0.01555	0.02139
Average:	0.82618	0.42194	0.87242	0.75491	0.87347	0.90964	0.89127	0.91828
Min:	0.79112	0.25900	0.83389	0.69712	0.84435	0.89420	0.85408	0.84225
Max:	0.85074	0.58100	0.91431	0.80256	0.89923	0.92399	0.91989	0.95141

6.3 Processing time

In order to answer Research Question 2, results for processing times for different model-hardware combination are presented in this section. To help estimate the processing time on mobile phones, results from Furseth and Granås' thesis [15] are used to find a suitable ratio to estimate the inference time. Five comparisons, with corresponding ratios, can be seen in Table 6.5. The table shows that the ratio between the laptop and the mobile inference times lie, for all measurements, within the range of 10.071 to 12.083, averaging at 11.059. This average ratio will be used in further comparisons, as a means to estimate the mobile processing times.

Table 6.5: Inference times on the laptop compared to the mobile phone times found by Furseth and Granås [15], in order to find an approximate ratio. The small untiled YOLOv5s is used with different input image sizes.

Image size [px]	Laptop [s]	Mobile [s]	Ratios
640	0.012	0.145	12.083
1024	0.028	0.297	10.607
1280	0.038	0.447	11.763
1920	0.079	0.851	10.772
2624	0.141	1.420	10.071
		Average:	11.059

Results from the measurement and estimation of processing times for IDUN, the laptop and the mobile phone can be seen in Table 6.6. The results clearly show that for both IDUN and the laptop, all model variation timings fall within the acceptable 5.85-second time requirement. IDUN's results all fall within the range from 0.011 to 1.170 seconds, and the laptop achieves a minimum time of 0.044 seconds and a maximum time of 4.060 seconds. For the estimated mobile results, however, the answer isn't as simple. As the requirement stands at 5.85 seconds, none of the large, tiled models are nearly fast enough on the hardware found in the mobile phone. In addition to this, the small tiled fusion model is estimated to use about 6.7 seconds, meaning it does not fall within the requirement either. All other models are estimated to successfully run on a mobile phone within the required time.

Table 6.6: Processing times for the different models on IDUN, the laptop and the mobile phone. The mobile phone processing times are italized, and are estimates as described in subsection 5.7.2.

Input	Partitioned	Base model	Hardware	Preprocessing [s]	Inference [s]	Postprocessing [s]	Total [s]
RGB	False	YOLOv5s	IDUN	0.001	0.008	0.002	0.011
			Laptop	0.003	0.039	0.003	0.044
			Mobile	*0.050*	*0.431*	*0.050*	*0.531*
		YOLOv5l6	IDUN	0.001	0.014	0.002	0.016
			Laptop	0.005	0.183	0.015	0.203
			Mobile	*0.050*	*2.024*	*0.050*	*2.124*
	True	YOLOv5s	IDUN	0.662	0.030	0.015	0.707
			Laptop	0.538	0.297	0.016	0.851
			Mobile	*1.000*	*3.285*	*0.100*	*4.385*
		YOLOv5l6	IDUN	0.662	0.098	0.015	0.775
			Laptop	0.541	2.244	0.017	2.802
			Mobile	*1.000*	*24.817*	*0.100*	*25.917*
RGB + IR (fusion)	False	YOLOv5s	IDUN	0.898	0.008	0.001	0.907
			Laptop	0.708	0.063	0.003	0.773
			Mobile	*1.000*	*0.697*	*0.050*	*1.747*
		YOLOv5l6	IDUN	0.898	0.022	0.001	0.921
			Laptop	0.711	0.338	0.013	1.061
			Mobile	*1.000*	*3.738*	*0.050*	*4.788*
	True	YOLOv5s	IDUN	0.985	0.050	0.009	1.044
			Laptop	0.803	0.488	0.015	1.306
			Mobile	*1.200*	*5.397*	*0.100*	*6.697*
		YOLOv5l6	IDUN	0.985	0.164	0.021	1.170
			Laptop	0.819	3.216	0.024	4.060
			Mobile	*1.200*	*35.567*	*0.100*	*36.867*
RGB + IR (4 channel)	False	YOLOv5s	IDUN	0.898	0.004	0.001	0.903
			Laptop	0.707	0.041	0.002	0.751
			Mobile	*1.000*	*0.453*	*0.050*	*1.503*
		YOLOv5l6	IDUN	0.898	0.015	0.001	0.913
			Laptop	0.709	0.185	0.003	0.897
			Mobile	*1.000*	*2.046*	*0.050*	*3.096*
	True	YOLOv5s	IDUN	0.985	0.032	0.009	1.025
			Laptop	0.803	0.296	0.016	1.115
			Mobile	*1.200*	*3.274*	*0.100*	*4.574*
		YOLOv5l6	IDUN	0.985	0.101	0.021	1.107
			Laptop	0.810	2.389	0.017	3.216
			Mobile	*1.200*	*26.421*	*0.100*	*27.721*

6.4 Colored and obscured sheep

As first presented in section 1.3 and further explained throughout chapter 5, Research Question 3 requires the examination of the models' performance on the different colors of sheep and on obscured and covered sheep.

6.4.1 Colored sheep

The results from the colored sheep case study can be seen in Table 6.7. It is worth noticing, as presented in section 5.1, that the number of white sheep in the dataset is far higher than for the other colors. From the numbers in the table, the models perform quite differently, and no model is the best at multiple colors of sheep. Tiled RGB with YOLOv5s performs best on white sheep with a recall of 0.943. For gray the small untiled fusion model perform the best with a recall of 0.884, while for the black sheep the best results are found using the small tiled fusion model with a recall of 0.926. Finally, the best model for finding brown sheep is the large tiled 4 channel model with a recall of 0.966, which also is the best color specific results found in the table.

Table 6.7: Recall compared for the differently colored sheep, from the 12 combinatorial runs when validated on the test dataset.

				Color recall			
				White sheep	**Gray sheep**	**Black sheep**	**Brown sheep**
Input	**Tiled**	**Base model**	**Conf. thresh.**	***Total: 690***	***Total: 181***	***Total: 54***	***Total: 59***
RGB	False	YOLOv5s	0.639	0.919	0.851	0.889	0.898
	False	YOLOv5l6	0.665	0.914	0.829	0.870	0.898
	True	YOLOv5s	0.725	**0.943**	0.867	0.889	0.949
	True	YOLOv5l6	0.778	0.917	0.856	0.870	0.932
RGB + IR (fusion)	False	YOLOv5s	0.557	0.906	**0.884**	0.852	0.932
	False	YOLOv5l6	0.436	0.923	0.862	0.815	0.898
	True	YOLOv5s	0.732	0.923	0.878	**0.926**	0.949
	True	YOLOv5l6	0.737	0.874	0.840	0.833	0.898
RGB + IR (4 channel)	False	YOLOv5s	0.598	0.917	0.856	0.889	0.915
	False	YOLOv5l6	0.588	0.910	0.856	0.778	0.881
	True	YOLOv5s	0.728	0.923	0.878	0.907	0.949
	True	YOLOv5l6	0.703	0.916	0.867	0.889	**0.966**

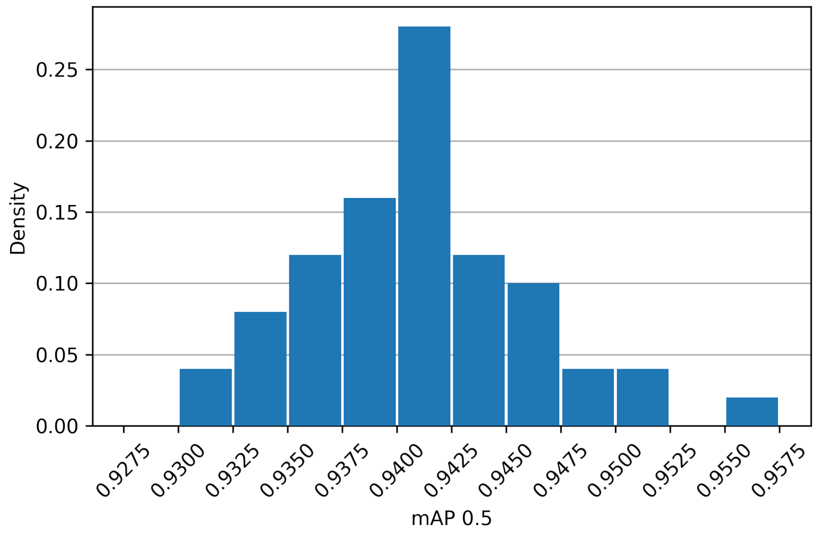
6.4.2 Obscured sheep

The results from the obscured sheep case study are shown in Table 6.8. During training and validation, most sheep are in the open. This is also true for the test dataset, as is seen in the header row; most sheep are not obscured or covered at all. The small tiled 4 channel model performs the best on the sheep found in the open with a recall of 0.974. For all three of the occluded categories, the small tiled RGB-model achieves the best results. The model achieves a recall of 0.368 for the completely obscured sheep, a recall of 0.818 for the partially obscured sheep and a recall of 0.684 for the partially covered sheep. However, the large, tiled 4-channel model also achieves a recall of 0.684 for the partially covered sheep.

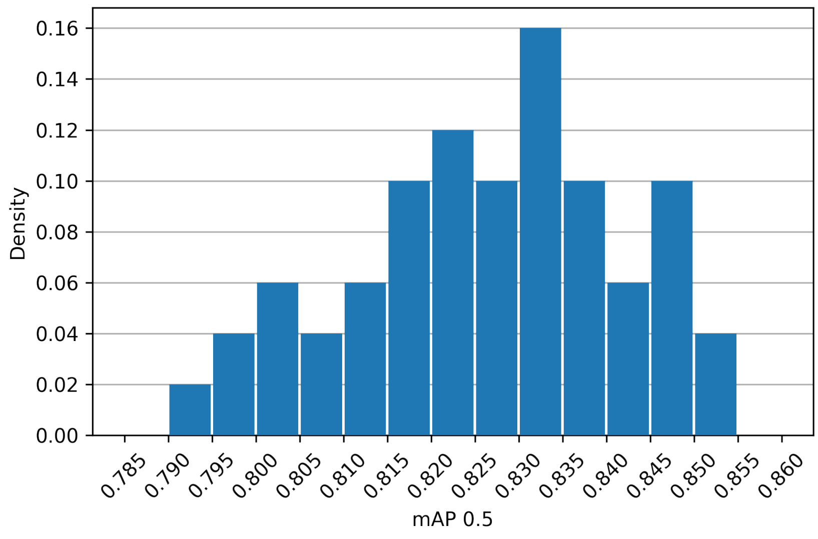
Note that the recall for the sheep which were out in the open are quite similar, and all fall within 0.952 to 0.974, while the recall for the occluded categories vary drastically. Recall for completely obscured sheep vary from 0.184 to 0.368. Partially obscured sheep all achieve a recall within the range of 0.621 to 0.818, while the recall for partially covered sheep varies from 0.447 to 0.684.

Table 6.8: Recall compared for different levels of occlusion, from the 12 combinatorial runs when validated on the test dataset.

				Occlusion recall			
				In the open	**Completely obscured**	**Partially obscured**	**Partially cover**
Input	**Tiled**	**Base model**	**Conf thresh.**	***Total: 841***	***Total: 38***	***Total: 66***	***Total: 38***
RGB	False	YOLOv5s	0.639	0.964	0.184	0.712	0.632
	False	YOLOv5l6	0.665	0.952	0.263	0.758	0.526
	True	YOLOv5s	0.725	0.973	**0.368**	**0.818**	**0.684**
	True	YOLOv5l6	0.778	0.970	0.211	0.712	0.500
RGB + IR (fusion)	False	YOLOv5s	0.557	0.966	0.289	0.636	0.553
	False	YOLOv5l6	0.436	0.960	0.316	0.742	0.579
	True	YOLOv5s	0.732	0.970	0.316	0.773	0.605
	True	YOLOv5l6	0.737	0.939	0.184	0.591	0.447
RGB + IR (4 channel)	False	YOLOv5s	0.598	0.969	0.237	0.697	0.526
	False	YOLOv5l6	0.588	0.955	0.289	0.621	0.579
	True	YOLOv5s	0.728	**0.974**	0.342	0.697	0.605
	True	YOLOv5l6	0.703	0.969	0.211	0.682	**0.684**



(a) mAP0.5 on the test dataset.



(b) mAP0.5 on the Klæbu dataset.

Figure 6.3: Histograms showing the distributions of mAP0.5 values from the variance probing experiment on both test and Klæbu datasets.

Chapter 7

Discussion

The discussion strives to assess the research questions, with the theses' goal in mind: To *"Develop, train and evaluate a model that can accurately and efficiently detect and count sheep using RGB and IR images taken by UAVs."*. The more in-depth research questions, also described in section 1.3, will be discussed and an attempt will be made to answer them. The results on their own can not answer these questions, as there is some degree of ambiguity to be found in them. This chapter will attempt to clear up this ambiguity, as well as consider what factors have been sources to unfair advantages and disadvantages when compared with other state-of-the-art models.

7.1 Experimental results

In order to answer Research Question 1, it is necessary to look thoroughly through the results on the test dataset and Klæbu dataset for the different models. When looking at the scatter plot for the test dataset in Figure 6.1a, the larger tiled models surprisingly perform worse than all of the other models, while also having the longest inference time. The tiled small models perform the best, an indication that tiling the image before running inference actually imposes an increase to mAP0.5. The model with the highest mAP0.5 is the small tiled 4 channel model, which also has a reasonable inference time. The small tiled fusion model also achieves the same mAP0.5, but with significant longer inference time. There is no significant indication that fusion or 4 channel performs any better than the RGB model.

However, looking at the scatter plot for the Klæbu dataset in Figure 6.1b, the spread is much larger than in the test set. This is not too surprising as the Klæbu dataset consists of images with quite different terrain to what the models were trained on. Thus, the models have not been trained specifically for terrains such as the ones found in Klæbu.

The larger tiled models, especially the fusion model, have the highest mAP0.5. All the 4 channel models lie on the pareto front, and gives the best mAP0.5 in the three groups with lowest inference. In all the four groups both fusion and 4 channel give higher mAP0.5 than the RGB model in their group. This is an indication that the use of the IR images gives an increase to mAP0.5. With regards to the RGB+IR models performing better on this dataset, that might be because images are taken in May with relatively freshly sheared sheep and cooler ground. Therefore the IR images have a higher contrast between the ground and the sheep, and it is easier for the models to utilize the extra information.

In Table 6.1, it can be seen that all models have a slight decrease in grid average precision compared to mAP0.5. On the other hand the opposite can be seen in Table 6.2, where the grid average precision sees a substantial increase compared to the mAP0.5 score. This might be due to the sheep in the test dataset being more spread out, while some of the images in the Klæbu set have clusters of sheep, due to the sheep gathering around feeding stations. The close proximity of many sheep makes it harder for the neural network when trying to predict correct bounding boxes for each individual sheep, while making it is easier for a grid based neural network to predict whether or not there are sheep inside a grid cell. The easier classification for the grid metric can also be seen when comparing the bounding box recall with the grid sheep recall. All models have a higher grid sheep recall than bounding box recall on both the test dataset andthe Klæbu dataset. This is probably because the metric classifies all sheep within a grid cell as found, even if only one sheep is found, while the bounding box metrics depend on correctly predicting each individual sheep to increase the recall score. While this is stricter, it will more precisely measure how well the model actually recognizes sheep. Still, for the sheep retrieval use case it will be sufficient to only find one sheep in an image, as the sheep farmer will have to travel to that destination to find the sheep anyway.

7.1.1 Normal distribution comparisons and their validity

When comparing the performance of different models, it is helpful to use statistical tools to describe the probabilities of one model truly being better than another. Assuming that the model results follow normal distributions, there are known probabilities that one model's mAP0.5 will be within n standard deviations (σ) of the distributions mean (μ), as explained in section 3.10. This is visualized in the scatter plots, using a ruler from the average result found in the variance probing experiment. On the test dataset, the small tiled models all achieve results larger than the base model's $\mu + 2\sigma$, which would be highly unlikely unless there is an actual increase in performance due to tiling. This allows us to discard the null hypothesis for these models. The same can be said when looking at the scatter plot from the Klæbu dataset, as it is extremely likely that the use of IR images as well as tiling gives an increase to the mAP0.5. Thus, the probing experiment helps prove, beyond the significance level of $\alpha = 4.6\%$, that the results in the main experiment are not mere coincidence, and that for all the models achieving results more extreme than $\mu \pm 2\sigma$ the null hypothesis can be discarded.

One potential improvement to this argumentation would be to train multiple models of each combination, and compare each combination's average scores. Due to the fact that the distribution of a mean is much tighter than the distribution of a single point, results would carry a clearer statistical significance [55]. It is also worth noting that because these models have not been variance probed, it is more difficult to statistically compare any two results, and results can only be directly compared with the base-model, which is the untiled large RGB-only model. By running e.g. 50 identical runs of each model, the models' metrics could be averaged and compared to each other with much higher certainty.

Another issue with this approach is that it assumes that the model result distributions are equal, and that they share the same standard deviation values. This means that if another model's metric distribution has a much larger standard deviation, it would have a higher chance of achieving more extreme results. It is, however, not probable that distributions have too high standard deviation values in these cases, as it is impossible for a model to achieve an mAP0.5 value of over 1.0. It is worth noting that as scores approach 1.0, distributions should get tighter and tighter, giving a lower standard deviation. This would make it even more unlikely for a value to stray as far from its distribution's mean.

The final issue worth noting about this discussion, is that it might not even be fair to assume that the mAP0.5 follows a normal distribution. Looking at the histograms in Figure 6.3, the mAP0.5 distribution for the test dataset seems to be centered and approximately normal distributed, but the values found for the Klæbu dataset clearly lean toward the right, and do not seem to follow a clear normal distribution.

In spite of all these issues, the standard deviation still gives a clear indicator to how much randomness there is within a distribution. Because of this, finding result scores higher than $\mu + 2\sigma$ quite definitely indicates that these modified models provide better results than the baseline large untiled RGB model.

7.1.2 Time restrictions

In order to try to give an answer to Research Question 2, it is necessary to discuss the results from the processing time experiment. When looking at the measurement and estimation of processing times shown in Table 6.6, most of the results look promising for the proposed use cases. It is not surprising that the mobile phone struggles to keep up with the other devices when processing the large tiled models. Backed by these results, the two possible solutions are still relevant, with the cloud solution offering more than fast enough processing for even the heaviest of models. A laptop should also manage quite well within the time restrictions. Still, the mobile phone does manage large untiled models and most smaller tiled models within the time restriction, and such a solution can therefore still be applicable.

The use of tiled models has the largest impact on processing time, as this introduces some preprocessing and postprocessing in addition to having 6 times longer inference time. The preprocessing of IR images introduces almost a second of preprocessing. When looking at the inference time for the models using IR images, the 4 channel model uses approximately the same time as the RGB model. On the other hand the fusion model uses significantly more time, due to the fact that it is running the backbone twice. The larger YOLOv5l6 also has an impact on the inference time compared to using the smaller YOLOv5s. It is important to note that a slight change on inference time on IDUN, will have a larger impact on the inference time on the laptop, and a significant impact on the mobile phone.

The basis for the set time restriction of 5.85 seconds is that it is the estimated time the drone uses between shots. This number is not based on empirical and experimental data, but rather on multiple assumptions. One of these assumptions is that the drone's acceleration and deceleration is constant and not related to its velocity. In the real world, it is much more plausible that the acceleration would start off high, but slowly approach zero as the drone nears its top speed. This is the nature of top speeds, which is the speed at which an object can no longer provide more force than the drag forces affecting it. Similarly, the deceleration would start off much higher than the acceleration, as the drag is now helping in the braking process, and end up at the same rate as the maximum of the acceleration as the velocity nears zero. This affects the estimation of travel time, but as the purpose of the estimation is to offer a plausible time requirement, the estimate is acceptable.

7.1.3 Sheep colors and occlusion

In order to answer Research Question 3, it is crucial to look at how the models perform on the different colors of sheep, as well as to what degree they can locate sheep that are obscured by vegetation and other obstacles. When looking at sheep color and partially hidden sheep there is one model standing out, namely the tiled small RGB model. Looking at the main experiment results, this is backed up by the model having the best bounding box recall, grid recall and grid sheep recall. This gives the model an advantage when looking at the sheep color and partially hidden sheep, as only the recall is measured. It is worth mentioning that the confidence threshold can be lowered for all models in order to achieve better results for recall, which is further discussed in subsection 7.2.4.

In Table 6.7, the trend is that the models perform better when finding white sheep than the other colors. All models struggle a bit with gray sheep, while some models excel at locating black or brown sheep. This might be due to the small amount of black and brown sheep in the dataset. All the datasets have a higher ratio of white sheep than the other colors, which probably makes the model quite good at finding them. Additionally, white wool stands more out from the ground, as long as there is not snow on the ground. The results do not indicate to which degree specifically colored sheep are more frequently occluded, so this could possibly be a source of error in these tables.

Recalls for sheep at different occlusion levels are shown in Table 6.8. From the table, it is apparent that the recall for sheep in the open is much higher than for the other categories, which can be easily explained by the necessary object more landmarks are visible in the images. Because of this, the main experimental results might contain more nuance than what is found in the table, given that the network scores so much better on non-occluded sheep. Had the main experiment been repeated on the dataset used by Sarwar et al. [16], which exclusively contained images of fenced sheep, the results would likely improve. However, this is not a practical change for this thesis, because in the use case of Norwegian sheep farmers, an even larger portion of the sheep are probably occluded than what is found in the datasets used for training, validation and testing in this thesis. This would therefore be disadvantageous to the models' performance on occluded sheep. This could be balanced out by having a higher percentage of occluded sheep in the training dataset. In this case, as mentioned, the test dataset was deliberately given more images with occluded sheep in order for the metrics to be more trustworthy. Some of these images contains sheep that are very difficult to spot, even for a human trying to label them. If it was desirable to force a model to predict the location of barely visible sheep, by lowering the confidence threshold, it is probable that the rate of false positives would also rise, as the model could no longer rely on all the landmarks of a full sheep.

7.2 Other sources of error

It is important to minimize the risk of errors during experiments. In some cases, the act of avoiding these errors is straight forward. In other cases, the sources of error are not as easily avoided or possibly not detected until the experiment is concluded. Some more sources of error are described below.

7.2.1 Empty tile labels

During tiling, it's quite possible for a bounding box to exist near the edge or corner of a tile, without there being a single pixel from the sheep in that tile. An example of this can be seen in Figure 5.7, where sheep 3 has a tiny label in tile T01, although there are no visible pixels from the sheep. This might make the network training more unpredictable, as the network will not know how to deal with the wrongful labels. One solution would be to manually look over every tiled image and remove erroneous labels, but considering that there are 6 tiles per image and about 1500 image pairs, this job would require hours of tedious work, without any guarantee that it will improve the results in any notable way.

7.2.2 Infrared quality

The first, most obvious problem with the infrared images is that they are so small. At a true resolution of 160 × 120 pixels, there is very little information in these images, and scaling them up does not remedy this. In spite of this obvious lack of quality, the infrared images clearly help the networks locate sheep. Therefore it could be interesting to see what a set of higher quality infrared images could do to the results.

Another issue when comparing the labels with the infrared images, is that the labels do not line up perfectly with the sheep. This misalignment might make it harder for the model to learn how to make use of the infrared images. One reason for this discrepancy is that the RGB image is captured before the infrared image, with a slight delay. This delay might allow the drone to move or drift slightly between captures, which would misalign the images. Another reason, looking at some image pairs, seems to be that the alignment preprocessing steps are imperfect, as many of the image pairs are misaligned in the same manner. It is doubtful that this misalignment happens due to drift from wind/movement, given how similar it is. These misalignments, combined with the fact that the labeling process only uses the RGB images, might explain the discrepancies found in the IR images when compared with the labels. Additionally, fine-tuning of the IR camera's temperature threshold might also yield better images for the network to process.

7.2.3 Dataset issues

There are obvious advantages in having a completely separate dataset from Klæbu. Firstly, when comparing results with those found in other theses, basing the work on the same dataset allows for direct comparison. Secondly, having a separate dataset with relatively unique data allows for more accurate measuring of model generality, by allowing testing of the model on previously unseen images. By this metric, the models seem to have generalized well enough, as the results on the Klæbu dataset are at an acceptable level, albeit achieving slightly lower scores than on the test dataset.

One problem with the datasets is that there might not be enough completely unique images. As presented in section 3.4, two important features of a good dataset is that it is large, and that it contains varied data. The problem with the 1500 images present in the dataset is that many of them are taken in series. In some cases, there are more than 50 images in a series, where the contents of these images sometimes vary only marginally. This can be seen in the dataset as the images are sometimes only rotated slightly, or the drone might be elevated to capture a slightly larger area or the operator flies the drone a few meters in one direction. All these small changes only give slightly different images, meaning having e.g. 50 images in a series are not as helpful to model generalization as 50 completely unique images. Still, when training, 50 similar images still help the network evolve more than one image.

Similarly to that problem, it might be problematic that the test dataset contains full series of images. The test-dataset is hand-curated to contain as varied data as possible. Complete series from the training dataset are picked to avoid training on almost identical images. This might cause the test dataset to become more biased than a dataset consisting of singular images would be. Another solution would be to pick single images from the training dataset. That approach would, however, let the models test and train on images from the same series, which would drastically lower the credibility of the results. Therefore, based on the available data, the chosen solution is deemed better.

Another important point, with regard to the quality of the dataset, is that the dataset still lacks a large portion of relevant images. The proposed usage areas for the system are in highland grazing areas, while a substantial portion of these images are taken in fenced areas. Additionally, some parts of the dataset are captured in the wrong seasons. An example of this is the Klæbu dataset, which is captured in May within the fenced area of a farmer in Klæbu. Weather reports for Klæbu in the days prior to and on the day the images were captured (May 10th 2020), show that the temperature ranged from around 0°C, to around 10°C, meaning the ground temperature was likely quite low [56]. Temperatures like this is not representative for the early autumn and might lead to better results for models utilizing IR images, for this particular dataset. Still, it is promising that the models can utilize the IR data and that it gives better performance, as the images from the autumn can be furthered improved with a better IR camera.

7.2.4 Confidence threshold selection

The F1 curve, as mentioned in subsection 3.8.2, is helpful as a tool to maximize both the recall and the precision metrics, by picking the confidence threshold which maximizes the F1 value. A problem with this confidence threshold selection, is that it neither maximizes precision nor recall on their own. It is possible to manipulate the confidence threshold in order to maximize one of the scores. Therefore, it does not make much sense to compare the recall score of one model with that of another, and the same is true for precision. In spite of this, these metrics do offer an indication of how well the model performs, and are easier to interpret for humans.

Another problem with this mathematical selection of confidence thresholds, is that it does not offer the user any input into what is important for them. For these use cases, it was presented in subsection 5.3.2 that recall is prioritized slightly over precision when training. The basis for this prioritization is that at the frequency the images are captured and analyzed, an operator would easily be able to sift through the false positives and discard them when necessary. This choice is, however, not reflected in the F1 approach to selecting a confidence threshold. Therefore it might be more appropriate to select a slightly lower confidence threshold, as that would give more predictions and higher recall. It is worth noting that some of the added predictions will be false positives leading to a lower precision. Another approach would be to select a static confidence threshold of 0.5. However, this would not improve the results' credibility, as the various models have a different understanding of what a confidence of 0.5 actually means.

7.3 Project boundaries

In chapter 4 and in chapter 5, the boundaries for the project are respectively defined and applied. Some of these boundaries are set due to defined solution requirements or physical limitations, while others are more arbitrary and could possibly be defined differently.

7.3.1 Necessary detection accuracy

Johannessen proposes that it is not crucial to precisely pin-point the location of every sheep, as she in her thesis develops a grid based detector, which only detect whether a sheep is present within a larger part of an image [14]. It is worth noting that the premise for Johannessen's proposal is that it could increase the overall performance of the model, by decreasing the inference time. A clear disadvantage of grid based detectors is that they make it harder to definitely and precisely tell what makes a sheep difficult to locate. With a grid based detector some difficult sheep standing next to simple ones might be found, while others are not found at all. Although grid based detection might locate more sheep during validation, measured using the grid based sheep recall, this thesis argues that this can be simply explained by the network suddenly only needing to detect a few sheep in each cluster, as sheep often graze in herds.

7.3.2 Emphasis on precision or recall

Furseth and Granås speculate that for the use case where an operator looks through the network's predictions, it is more important to achieve high precision than high recall [15]. They do not mean that recall is irrelevant, but argue that in order to decrease the rate of false positives, the recall needs to be forsaken in favor of precision.

The basis for their argumentation is that inference would be run every 0.5 seconds, which is not true for the proposed use case in this thesis. Estimates show that the detection will only need to be run once every 65 meters, at an interval of approximately of 5.85 seconds, as calculated in section 5.7. This should allow for a slightly improved recall, at the cost of lowered precision, as it is believed that an operator would be able to quickly dismiss false positives, while the network finds more sheep in the images. This is a valid compromise because it is important for the farmer to find all their sheep after the summer, although it might require slightly more work.

7.4 Comparison with K. Johannessen

Johannessen's results [14] work as good benchmarks for the models discussed in this thesis, and her models can be described as being the state-of-the-art in sheep recognition in UAV imagery of rangelands. It is worth noting, as mentioned in subsection 3.8.4, that the bounding box prediction metrics produced by a YOLOv5 neural network need transforming before being comparable to Johannessen's grid metrics. It is difficult to say to what degree this difference and transformation affects the comparability of the results. Seemingly, the results are within range of each other. By logically examining the process used, one might argue that the post processed bounding box detector works similarly enough to the direct grid detector, as the bounding boxes are transformed into grid detections.

The Klæbu dataset, in Johannessen's thesis referred to as T1, works as a benchmark for the results, as it ensures both sets of models have the same challenge. Using different test datasets would lead to misleading results due to one dataset being easier than the other, while using the same dataset allows the models' results to be directly compared. As Johannessen documents the grid AP metric on the Klæbu dataset, this will be the metric used in comparisons. By reading from Johannessen's Appendix 6, her three best grid AP scores on the Klæbu dataset are 0.945, 0.945 and 0.944, for three quite different models. Those models had inference times of 0.586, 0.349 and 0.337 seconds respectively. By comparison, the best models presented in this thesis achieved grid AP scores of 0.974, 0.953 and 0.947 on the Klæbu dataset, and had inference times of 0.165, 0.037 and 0.094 seconds, respectively. By this comparison, it is possible that the modifications made to YOLOv5 presented in this thesis has similar or better results than the state-of-the-art for machine learning-based sheep recognition in UAV imagery. Although the grid AP scores are quite comparable between the two theses, the inference times are not. It is difficult to directly compare inference times as Johannessen's results are measured on two NVIDIA GTX 1080 GPUs, while this thesis' results are measured on an NVIDIA A100 GPU. Therefore no attempt will be made to further compare the two metrics.

One factor which contributes to the improved results in this thesis, is the larger size and variety of the training data. Johannessen's models had access to 515 relevant image pairs, while this thesis has worked with 1591 relevant image pairs, allowing the models to generalize even more. Another important point to mention is that while Johannessen's thesis focused on applying fusion of IR and RGB data to a simpler network, this thesis applies the concept to one of the current state-of-the-art object detection networks.

7.5 Application in use cases

When it comes to localization of sheep there are two separate tasks requiring modernization, as described in section 4.1. The first task is the search and retrieval of sheep at the end of the summer season, with the intent of gathering all sheep back from the grazing areas. The second task is the weekly inspections of the farmer's herd, which is required by the government in order for the farmer to receive subsidies and for the welfare of the sheep.

When evaluating if the results are applicable to the task of locating and retrieving sheep, an important point is that the information is both trustworthy and received in a timely manner. The timeliness is especially important for the round-up use case, as mentioned in section 4.4. As the processing time results show that the mobile solution can not run the models with the best results within the time requirement, this solution is not as relevant to the use case. Although the server solution requires costly cellular data to transfer the images to the server for processing, it might also cut costs by not requiring the farmer to invest in a laptop computer with a dedicated GPU. This trade-off goes both ways, as for some farmers, the investment of a slightly more costly laptop computer might be preferable to the monthly costs of a cloud service provider subscriptions. Another advantage of the laptop solution is that it does not require a cellular connection, which also saves on the cellular subscription. Instead, only a connection to the drone is required to transfer the image data from drone to laptop, allowing it to perform the processing and locate the sheep locally. In any regard, both solutions are considered useful until further cost analysis is performed. It is worth noting that the cloud solution is more dynamic and in exchange for a steep increase in cloud costs, multiple server instances could be used simultaneously, which would allow for the usage of larger models, larger image sizes and more frequent image capture.

As there is no timely dimension when performing weekly inspections, the cloud based solution may be applied in an asynchronous manner, meaning that all the images may be captured and brought back to a wired connection, and then be uploaded to the cloud service from there. This solution is worth discussing because it solves problems found in both the mobile and the cellular cloud based solutions: It comes at a reduced price, as it does not depend on cellular usage, as well as being a compute-heavy cloud solution for the weekly required workload. The farmer will be able to get the relevant images containing sheep and therefore be able to inspect the welfare of the sheep without having to travel out to find them. Although the count of the sheep by using the proposed solution might not find all the sheep, it might give a good enough estimate, as it is difficult for the farmer to get a complete overview of his herd during normal inspections as well.

Chapter 8

Conclusion

This thesis discusses multiple variations of the YOLOv5 object detection network architecture, aimed at automatically detecting sheep in RGB and IR imagery, which will be captured by an autonomous drone. The image processing and machine learning inference is designed to run on either a remote server or on a mobile device. In section 1.3, one goal and three research questions were posed to drive the thesis forward. These research questions are discussed in chapter 7, based on the results presented in chapter 6 and will be concluded upon in this chapter.

Research Question 1 asks *"How do IR data usage, tiling and model size variations affect the accuracy of the model?"* Through statistically proof study these model variations have been tested, conclusions can begin to be drawn for each modification. Beginning with the usage of IR data, the results are not clear between the test dataset and the Klæbu dataset. Still, as the Klæbu results show such a clear improvement when using IR data, the conclusion is that the usage of IR data improves model accuracy when the IR image provides valuable information. The results do not warrant a conclusive comparison between the fusion model and the 4-channel model, as further discussed in subsection 7.1.1. Secondly comes the usage of tiling as a pre- and postprocessing step. Although there seems to be a slight improvement when utilizing tiling, there is a discrepancy when it comes to the large, tiled models on the test dataset, which somehow seem to perform worse than their untiled counterparts. Based on this, no conclusion can be drawn as to how tiling affects the accuracy of the model. Lastly, comparing the results between the small and the large model sizes, the Klæbu results show a large decline in mAP0.5 when using a small RGB model compared to the default large RGB model. This could warrant the dismissal of the null hypothesis, but as the results for the test dataset seem to show the complete opposite situation, the results are too conflicting to force the dismissal of the null hypothesis. Because of this, no scientific conclusion can be drawn, but looking at the data, the trend seems to be that larger models perform slightly better than the smaller ones.

Research Question 2 asks *"How do IR data usage, tiling and model size variations affect the processing time of images?"* As the inference times' standard deviation is relatively small, conclusions will be drawn from visible trends, and not from statistical analysis using a known distribution as a baseline. Beginning again with the usage of IR data, the two model variations seem to exhibit different properties. The fusion model, in which two separate backbones are processed, clearly has the highest inference time. The 4-channel model however, closely resembles the RGB-only model when it comes to inference time, which is believable as the only difference between the two architectures is the model's first layer. As a general conclusion, the usage of IR data need not result in a higher inference time, but this depends on the implementation and architecture. Comparing tiled and untiled models' inference times is a simple task, as the differences are both easily noticeable and explainable. As tiling requires the model to process 6 smaller sub-images instead of one large image, the processing times increase sixfold. There is no doubt that using tiling is a costly process, and that it should only be applied whenever inference time is not an issue. Finally, comparing inference times of small and large models, another clear trend is visible. The large models require much more processing, so there is little surprise in the results showing the smaller models achieving significantly lower inference times. As the drop in accuracy is not as significant and clear, smaller models might be a good option on lighter hardware.

Research Question 3 asks *"How well can the models detect sheep of varying colors and sheep which are partially hidden by vegetation?"* For colors, there does not seem to be any decisive trends showing that the models have difficulties locating any specific color of sheep. The presented hypothesis was that the results would favor white sheep, but in general the models locate all sheep fairly evenly, independent of the sheep's colors. Sheep occlusion results are, unsurprisingly, more varied. The more of a sheep that is visible in the image, the more likely it is that the model will detect it. Completely obscured sheep which are hard even for humans to notice, are rarely detected, and this is true for all models. Although some model variations perform better than others on the difficult classes, the general conclusion is that the results for obscured and covered sheep are so poor that practical application might not reach an acceptable level without better training data.

The goal of the thesis was to *"Develop, train and evaluate a model that can accurately and efficiently detect and count sheep using RGB and IR images taken by UAVs."* Given that this thesis presents results of mAP0.5 values up to 96%, and achieves performance scores that are on par with or better than those of previous state-of-the-art sheep detectors, the goal is reached. In spite of the great results, there is still research to be done. The most relevant research going forward, will be to consider practical challenges with the proposed system, as well as to consider untried approaches. These ideas and more, are discussed in chapter 9.

Chapter 9

Future work

The proposed points for future work attempts to address the nuances of possible areas of research which would help the project progress. This includes using a another type of drone, higher quality cameras, applying techniques to track sheep movement, attempting to use altitude data, as well as attempting a multi-camera setup for the drone or testing to see if ensemble learning would yield better results than those achieved in this thesis.

9.1 Winged drone

As the real world application of this project could be multiple times faster using a winged drone, which moves while capturing imagery, one imperative future task is to collect imagery from such a set-up. The collected data should then be applied through a network like the ones described in this thesis, to see if the necessary flight height and speed affects the results in any significant way. There would probably also be other time constraints for such a solution.

9.2 Higher quality IR imagery

The IR images used in this thesis have a base resolution of 160 × 120 pixels. Compared to the RGB images' 4056 × 3040 pixels, the IR imagery does not hold very much data. Another issue with the IR images, or more precisely; with the RGB and IR image pairs, is that they are not always properly aligned. Although it is hard to precisely address what happens inside a trained neural network during inference, it might be safe to assume that "messy" and unaligned data does not lead to good results. In the future, data collection should be done with an IR camera that offers higher resolution imagery, and the RGB and IR images must be properly aligned so that the network can be given "tidier" data.

9.3 Tracking sheep between images

It could be an interesting topic to track which sheep is which sheep when flying over an area and taking images with some overlap. As there will be some seconds in between image capture, they are probably not in the exact same position. This would make it possible to count the sheep with a higher precision if the images overlap. In addition, a harder challenge is to be able to track the same sheep when the UAV is on its way back on the fold, as multiple minutes might have passed since the neighboring image was taken, and the flock of sheep might have moved into the uncaptured area. Still, this is only a problem to get correct counting, for the simpler task of solely detecting sheep this is not a problem.

9.4 Using altitude data

In the dataset used in this thesis, the images have been taken while the drone was at different altitudes. A possible issue with this, is that the sheep in the images change relative size. A sheep being 20 pixels long and another being 80 pixels long, might be an issue for the neural network to handle. YOLOv5, utilizing anchor boxes of different sizes in its inference, should be a good choice of model for detecting objects of varying sizes. Still, the height at which the drone is flying could be a good data point to feed into the network, and attempting this could yield interesting results.

9.5 Multiple camera setup

The premise for this thesis, and its datasets, is that the images are taken straight down. This makes the images fairly predictable, and the drones direction (north, east, south, west) does not matter too much, as the neural network detects objects in any orientation. However, given the forested nature of many Norwegian grazing areas, capturing imagery from above might not be sufficient. If a sheep is standing underneath a spruce's thick branches, the network will have a difficult task trying to locate the sheep. To remedy this, it could be interesting to see if a camera pointed at an angle would prove better at locating these occluded sheep, as it could possibly "look" underneath the tree's branches. By mounting multiple cameras to the drone, it would be possible to apply both the "straight-down"-solution, as well as stitching the images together and possibly training/inferring on a large panorama of these cameras' images.

9.6 Ensemble learning

As the models trained is quite fast, it could be interesting to see if it is possible to achieve better results if multiple of them are used together, as in ensemble learning. In practice multiple models would detect sheep on their own, and then all detections would be combined using NMS, or something similar, before calculating the results. By using the different variations proposed in this thesis one would most likely achieve some performance increase as the different models have slight differences in which sheep they detect correctly. Although by utilizing completely different architectures one would very likely achieve an even higher performance increase.

Bibliography

[1] I. Nygård and S. Vittersø, "Gjenfinning av sau på beite, ved hjelp av drone og djup læring," Specialization project, Department of Computer Science, Norwegian University of Science and Technology, Trondheim, 2021.

[2] G. Austrheim, E. J. Solberg, A. Mysterud, M. Daverdin, and R. Andersen, "Hjortedyr og husdyr på beite i norsk utmark i perioden 1949–1999," *NTNU Vitenskapsmuseet - Rapport zoologisk*, 2008.

[3] Y. Rekdal, M. Angeloff, E. Skurdal, F. Avdem, V. Tømmerberg, and T. S. Tollersrud, *Utmarksbeite til sau*. Nortura, 2019. [Online]. Available: `https://www.geno.no/contentassets/68e6876772f147fe8bbecf28c473044f/utmarksbeite_sau_web.pdf`.

[4] Landbruksdirektoratet, *Pt-900 antallstatistikk*, `https://ldir.statistikkdata.no/pt-900_del2_2021_land.html`, [Retrieved 22. February 2022], 2021.

[5] Landbruks- og matdepartementet, *Forskrift om velferd for småfe*, `https://lovdata.no/forskrift/2005-02-18-160/\T1\textsection19`, [Retrieved 15. March 2022], 2005.

[6] Rovbase, *Erstatning for sau*, `https://rovbase.no/erstatning/sau`, [Retrieved 2. March 2022], 2021.

[7] Dyrebeskyttelsen, *Fanesak tap av sau på beite*, `https://www.dyrebeskyttelsen.no/tap-sau-pa-beite/`, [Retrieved 2. March 2022], 2020.

[8] Smartbjella, *Smartbjella website*, `https://smartbjella.no/butikk/`, [Retrieved 3. March 2022].

[9] Findmy, *Findmy website*, `https://www.findmy.no/nb/produkt`, [Retrieved 3. March 2022].

[10] Telespor, *Telespor website*, `https://nettbutikk.telespor.no/`, [Retrieved 3. March 2022].

[11] O. Flaten and L. Rønning, *Best på sau – faktorer som påvirker økonomisk resultat i saueholdet*. Norsk institutt for landbruksøkonomisk forskning, 2011. [Online]. Available: `https://nibio.brage.unit.no/nibio-xmlui/bitstream/handle/11250/2460578/NILF-Rapport-2011-03.pdf?sequence=2&isAllowed=y`.

[12] A. S. Haugset and G. Nossum, "Erfaringer med bruk av elektronisk overvåkningsutstyr på beitedyr," Trøndelag Forskning og Utvikling, 2010. [Online]. Available: `https://www.bondelaget.no/getfile.php/13117498-1310042486/MMA/Bilder%5C%20fylker/Nord%5C%20-%5C%20Tr%5C%C3%5C%B8ndelag/Dokumenter/Radibojeller%5C%20notat%5C%202010.pdf`.

[13] J. H. Muribø, "Locating sheep with yolov3," Master's Thesis, Department of Computer Science, Norwegian University of Science and Technology, Trondheim, 2019. [Online]. Available: `http://hdl.handle.net/11250/2619041`.

[14] K. M. Johannessen, "Towards improved sheep roundup - using deep learning-based detection on multichannel rgb and infrared uav imagery," Master's Thesis, Department of Civil, Environmental Engineering, Norwegian University of Science, and Technology, Trondheim, 2020. DOI: `no.ntnu:inspera:55924330:23111610`. [Online]. Available: `https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2779322`.

[15] O. K. Furseth and A. O. Granås, "Real-time sheep detection - improving retrieval of free-ranging sheep using deep learning-based detection on drone imagery running on mobile devices," Master's Thesis, Department of Computer Science, Norwegian University of Science and Technology, Trondheim, 2021. [Online]. Available: https://hdl.handle.net/11250/2834578.

[16] F. Sarwar, A. Griffin, S. U. Rehman, and T. Pasang, "Detecting sheep in uav images," *Computers and Electronics in Agriculture*, vol. 187, p. 106 219, 2021, ISSN: 0168-1699. DOI: https://doi.org/10.1016/j.compag.2021.106219. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169921002362.

[17] C. C. Aggarwal, *Neural Networks and Deep Learning*. Springer Cham, 2018. DOI: 10.1007/978-3-319-94463-0.

[18] IBM, *Neural networks*, https://www.ibm.com/cloud/learn/neural-networks, [Retrieved 22. February 2022], 2021.

[19] G. Dreyfus, "Neural networks: An overview," *Neural networks*, pp. 1–83, 2005.

[20] M. A. Nielsen, *Neural networks and deep learning*, 2018. [Online]. Available: http://www.neuralnetworksanddeeplearning.com/.

[21] L. Prechelt, "Early stopping — but when?" In *Neural Networks: Tricks of the Trade: Second Edition*, G. Montavon, G. B. Orr, and K.-R. Müller, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 53–67, ISBN: 978-3-642-35289-8. DOI: 10.1007/978-3-642-35289-8_5. [Online]. Available: https://doi.org/10.1007/978-3-642-35289-8_5.

[22] IBM, *Convolutional neural networks*, https://www.ibm.com/cloud/learn/convolutional-neural-networks, [Retrieved 22. February 2022], 2021.

[23] H. Hakim and A. Fadhil, "Survey: Convolution neural networks in object detection," *Journal of Physics: Conference Series*, vol. 1804, no. 1, p. 012 095, Feb. 2021. DOI: 10.1088/1742-6596/1804/1/012095. [Online]. Available: https://doi.org/10.1088/1742-6596/1804/1/012095.

[24] X. Jiang, A. Hadid, Y. Pang, E. Granger, and X. Feng, *Deep Learning in Object Detection and Recognition*. Springer Singapore, 2019. DOI: 10.1007/978-981-10-5152-4.

[25] J. Brownlee, *A gentle introduction to object recognition with deep learning*, https://www.machinelearningmastery.com/object-recognition-with-deep-learning/, [Retrieved 23. February 2022], 2019.

[26] Ultralytics, *Yolov5*, 2021. DOI: 10.5281/zenodo.4679653.

[27] Ultralytics, *Yolov5*, https://github.com/ultralytics/yolov5, [Retrieved 22. February 2022].

[28] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, *Yolov4: Optimal speed and accuracy of object detection*, 2020. arXiv: 2004.10934 [cs.CV].

[29] C. Supeshala, *Yolo v4 or yolo v5 or pp-yolo?* https://towardsdatascience.com/dad8e40f7109, [Retrieved 19. February 2022], 2020.

[30] J. Redmon and A. Farhadi, *Yolov3: An incremental improvement*, 2018. arXiv: 1804.02767 [cs.CV].

[31] G. Jocher, *Tips for best training results*, https://github.com/ultralytics/yolov5/wiki/Tips-for-Best-Training-Results, [Retrieved 20. April 2022], 2022.

[32] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, p. 60, Jul. 2019, ISSN: 2196-1115. DOI: 10.1186/s40537-019-0197-0. [Online]. Available: https://doi.org/10.1186/s40537-019-0197-0.

[33] F. Dadboud, V. Patel, V. Mehta, M. Bolic, and I. Mantegh, "Single-stage uav detection and classification with yolov5: Mosaic data augmentation and panet," in *2021 17th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2021, pp. 1–8. DOI: 10.1109/AVSS52988.2021.9663841.

[34] G. A. Reina, R. Panchumarthy, S. P. Thakur, A. Bastidas, and S. Bakas, "Systematic evaluation of image tiling adverse effects on deep learning semantic segmentation," *Frontiers in Neuroscience*, vol. 14, p. 65, 2020, ISSN: 1662-453X. DOI: 10.3389/fnins.2020.00065. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2020.00065.

[35] f. Unel, B. Ozkalayci, and C. Cigla, "The power of tiling for small object detection," Jun. 2019. DOI: 10.1109/CVPRW.2019.00084.

[36] J. Hsieh and A. Chia, "Object detection with partial occlusion based on a deformable parts-based model," 2010. [Online]. Available: http://cs229.stanford.edu/proj2010/HsiehChia-ObjectDetectionWithPartialOcclusion.pdf.

[37] T. Ophoff, K. Van Beeck, and T. Goedemé, "Exploring rgb+depth fusion for real-time object detection," *Sensors*, vol. 19, no. 4, 2019, ISSN: 1424-8220. DOI: 10.3390/s19040866. [Online]. Available: https://www.mdpi.com/1424-8220/19/4/866.

[38] R. Padilla, W. L. Passos, T. L. B. Dias, S. L. Netto, and E. A. B. da Silva, "A comparative analysis of object detection metrics with a companion open-source toolkit," *Electronics*, vol. 10, no. 3, 2021, ISSN: 2079-9292. DOI: 10.3390/electronics10030279. [Online]. Available: https://www.mdpi.com/2079-9292/10/3/279.

[39] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, *Microsoft coco: Common objects in context*, 2015. arXiv: 1405.0312 [cs.CV].

[40] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun. 2010, ISSN: 1573-1405. DOI: 10.1007/s11263-009-0275-4. [Online]. Available: https://doi.org/10.1007/s11263-009-0275-4.

[41] S. K, *Non-maximum suppression (nms)*, https://towardsdatascience.com/non-maximum-suppression-nms-93ce178e177c, [Retrieved 20. April 2022], 2019.

[42] M. James, *Statistical significance*, https://www.investopedia.com/terms/s/statistical-significance.asp, [Retrieved 27. April 2022], 2021.

[43] B. Wesolowski and D. Musselwhite Thompson, "Normal distribution," Jan. 2018. DOI: 10.4135/9781506326139.n476.

[44] M. Galarnyk, *Explaining the 68-95-99.7 rule for a normal distribution*, https://towardsdatascience.com/b7b7cbf760c2, [Retrieved 3. March 2022], 2018.

[45] D. Carr, *How to calculate field of view in photography*, https://shuttermuse.com/calculate-field-of-view-camera-lens/, [Retrieved 18. March 2022], 2016.

[46] S.-O. Hvasshovd, *Droner og sau og litt til !! anvendelser og muligheter*, https://www.statsforvalteren.no/contentassets/cbf122460efa4e37a051c17c07fade0d/droner-buskerud-2017.pdf, [Retrieved 2. May 2022], 2017.

[47] A. Buffi, P. Nepa, and R. Cioni, "Sarfid on drone: Drone-based uhf-rfid tag localization," in *2017 IEEE International Conference on RFID Technology Application (RFID-TA)*, 2017, pp. 40–44. DOI: 10.1109/RFID-TA.2017.8098872.

[48] T. Freed, V. C. Carson, and K. H. Doerr, "Optimizing a rfid-uav cattle search tour," *International Journal of RF Technologies*, vol. 11, pp. 127–141, 2020, 2, ISSN: 1754-5749. DOI: 10.3233/RFT-180163. [Online]. Available: https://doi.org/10.3233/RFT-180163.

[49] DJI, *Dji mavic 2 enterprise series*, `https://www.dji.com/no/mavic-2-enterprise/specs`, [Retrieved 3. March 2022].

[50] M. Själander, M. Jahre, G. Tufte, and N. Reissmann, "EPIC: An Energy-Efficient, High-Performance GPGPU Computing Research Infrastructure," *arXiv:1912.05848 [cs]*, Dec. 2019. arXiv: `1912.05848 [cs]`.

[51] A. Thakur, *What's the optimal batch size to train a neural network?* `https://wandb.ai/ayush-thakur/dl-question-bank/reports/What-s-the-Optimal-Batch-Size-to-Train-a-Neural-Network---VmlldzoyMDkyNDU`, [Retrieved 17. March 2022], 2022.

[52] J. Solawetz, *The train, validation, test split and why you need it*, `https://blog.roboflow.com/train-test-split/`, [Retrieved 23. February 2022], 2020.

[53] X. Bouthillier, P. Delaunay, M. Bronzi, A. Trofimov, B. Nichyporuk, J. Szeto, N. Sepah, E. Raff, K. Madan, V. Voleti, S. E. Kahou, V. Michalski, D. Serdyuk, T. Arbel, C. Pal, G. Varoquaux, and P. Vincent, "Accounting for variance in machine learning benchmarks," *CoRR*, 2021. arXiv: `2103.03098`. [Online]. Available: `https://arxiv.org/abs/2103.03098`.

[54] J. Dsouza, *What is a gpu and do you need one in deep learning?* `https://towardsdatascience.com/718b9597aa0d`, [Retrieved 29. March 2022], 2020.

[55] Shafer and Zhang, "The sampling distribution of the sample mean," in *Introductory Statistics*. LibreTexts, Jan. 2021, ch. 6.2. [Online]. Available: `https://stats.libretexts.org/Bookshelves/Introductory_Statistics/Book%5C%3A_Introductory_Statistics_(Shafer_and_Zhang)/06%5C%3A_Sampling_Distributions/6.02%5C%3A_The_Sampling_Distribution_of_the_Sample_Mean`.

[56] Time and date, *Været som var i klæbu, norge — mai 2020*, `https://www.timeanddate.no/vaer/@3149954/siste-uke?month=5&year=2020`, [Retrieved 21. April 2022], 2020.

Appendix A

Complete variance results

The variance probing, as described in section 5.5, consists of 50 training runs. The metrics for these 50 models are found below, including one table showing the validation results when run on the test dataset, and one for the separate Klæbu dataset. As mentioned, these runs were run on an RGB-only YOLOv5l6 model without tiled images.

A.1 Test dataset

Table A.1: Variance probing results on the test dataset.

	Bounding box metrics				Grid metrics			
Run	**mAP0.5**	**Conf. thr.**	**Recall**	**Precision**	**AP**	**Precision**	**Recall**	**Sheep recall**
1	0.943	0.588	0.940	0.897	0.921	0.953	0.921	0.955
2	0.940	0.573	0.959	0.901	0.915	0.972	0.908	0.943
3	0.941	0.629	0.935	0.880	0.905	0.946	0.903	0.942
4	0.947	0.656	0.958	0.895	0.917	0.965	0.912	0.948
5	0.940	0.668	0.943	0.876	0.905	0.961	0.901	0.940
6	0.944	0.676	0.970	0.869	0.912	0.977	0.907	0.942
7	0.941	0.696	0.970	0.885	0.898	0.964	0.891	0.930
8	0.944	0.662	0.957	0.889	0.918	0.958	0.915	0.950
9	0.935	0.615	0.931	0.887	0.909	0.943	0.908	0.945
10	0.937	0.644	0.942	0.893	0.921	0.944	0.921	0.953
11	0.939	0.642	0.938	0.896	0.920	0.949	0.917	0.951
12	0.942	0.758	0.973	0.874	0.912	0.979	0.908	0.945
13	0.944	0.677	0.951	0.892	0.920	0.958	0.917	0.952
14	0.938	0.576	0.929	0.888	0.902	0.943	0.903	0.939
15	0.945	0.516	0.936	0.902	0.919	0.939	0.915	0.950
16	0.944	0.702	0.956	0.872	0.905	0.961	0.903	0.941
17	0.933	0.657	0.961	0.861	0.898	0.958	0.894	0.934
18	0.938	0.635	0.927	0.886	0.905	0.948	0.901	0.938
19	0.948	0.547	0.939	0.902	0.925	0.953	0.922	0.953
20	0.942	0.647	0.942	0.890	0.912	0.952	0.908	0.945
21	0.933	0.615	0.950	0.864	0.914	0.954	0.915	0.951
22	0.937	0.624	0.967	0.874	0.891	0.973	0.885	0.930
23	0.937	0.705	0.951	0.879	0.912	0.959	0.910	0.945
24	0.955	0.572	0.953	0.907	0.932	0.955	0.928	0.959
25	0.940	0.726	0.954	0.864	0.898	0.955	0.892	0.935
26	0.941	0.640	0.942	0.897	0.923	0.947	0.922	0.954
27	0.935	0.679	0.935	0.877	0.909	0.943	0.903	0.939
28	0.947	0.588	0.945	0.903	0.922	0.949	0.922	0.953
29	0.942	0.701	0.956	0.885	0.904	0.970	0.898	0.937
30	0.947	0.632	0.949	0.897	0.926	0.949	0.924	0.955
31	0.940	0.665	0.963	0.886	0.900	0.955	0.901	0.938
32	0.939	0.556	0.962	0.890	0.910	0.961	0.908	0.943
33	0.936	0.642	0.936	0.879	0.899	0.941	0.899	0.938
34	0.934	0.657	0.935	0.879	0.913	0.952	0.910	0.945
35	0.930	0.671	0.951	0.875	0.919	0.961	0.919	0.953
36	0.931	0.582	0.921	0.882	0.906	0.933	0.903	0.941
37	0.938	0.596	0.947	0.882	0.900	0.943	0.898	0.934
38	0.940	0.657	0.947	0.890	0.916	0.954	0.914	0.948
39	0.940	0.714	0.943	0.871	0.913	0.943	0.910	0.946
40	0.937	0.646	0.931	0.894	0.918	0.940	0.919	0.952
41	0.941	0.685	0.951	0.877	0.898	0.955	0.891	0.930
42	0.938	0.671	0.954	0.867	0.899	0.958	0.892	0.933
43	0.952	0.726	0.973	0.890	0.912	0.972	0.907	0.943
44	0.947	0.607	0.934	0.904	0.936	0.946	0.933	0.961
45	0.950	0.671	0.953	0.895	0.932	0.965	0.928	0.961
46	0.942	0.583	0.948	0.883	0.897	0.958	0.891	0.930
47	0.942	0.669	0.952	0.883	0.905	0.952	0.901	0.937
48	0.939	0.649	0.935	0.888	0.919	0.936	0.922	0.954
49	0.948	0.703	0.969	0.893	0.913	0.977	0.910	0.945
50	0.947	0.673	0.955	0.887	0.908	0.957	0.905	0.943
Variance:	2.71E-05	2.59E-03	1.65E-04	1.30E-04	1.04E-04	1.21E-04	1.25E-04	6.97E-05
Std. dev.	0.00520	0.05089	0.01283	0.01142	0.01018	0.01100	0.01116	0.00835
Average:	0.94111	0.64538	0.94836	0.88560	0.91163	0.95466	0.90875	0.94461
Min:	0.93038	0.51600	0.92112	0.86063	0.89129	0.93260	0.88536	0.92981
Max:	0.95545	0.75800	0.97330	0.90743	0.93558	0.97909	0.93298	0.96101

A.2 Klæbu dataset

Table A.2: Variance probing results on the Klæbu dataset.

	Bounding box metrics				Grid metrics			
Run	**mAP0.5**	**Conf. thr.**	**Recall**	**Precision**	**AP**	**Precision**	**Recall**	**Sheep recall**
1	0.831	0.443	0.871	0.758	0.858	0.904	0.871	0.925
2	0.802	0.538	0.914	0.697	0.879	0.924	0.887	0.842
3	0.833	0.340	0.890	0.757	0.872	0.920	0.887	0.933
4	0.824	0.444	0.888	0.744	0.891	0.917	0.906	0.915
5	0.804	0.494	0.856	0.736	0.853	0.912	0.876	0.904
6	0.834	0.509	0.910	0.750	0.862	0.924	0.868	0.905
7	0.837	0.436	0.885	0.742	0.871	0.911	0.896	0.929
8	0.818	0.581	0.890	0.721	0.855	0.915	0.864	0.885
9	0.838	0.398	0.864	0.756	0.888	0.903	0.893	0.921
10	0.833	0.543	0.882	0.754	0.877	0.907	0.890	0.901
11	0.837	0.358	0.840	0.798	0.899	0.908	0.920	0.951
12	0.823	0.427	0.852	0.756	0.866	0.896	0.888	0.916
13	0.813	0.500	0.878	0.734	0.850	0.913	0.867	0.910
14	0.817	0.471	0.881	0.738	0.882	0.906	0.886	0.901
15	0.828	0.354	0.863	0.756	0.873	0.902	0.898	0.919
16	0.795	0.259	0.834	0.724	0.849	0.898	0.871	0.908
17	0.816	0.389	0.876	0.759	0.868	0.900	0.890	0.901
18	0.838	0.355	0.868	0.765	0.893	0.918	0.903	0.913
19	0.827	0.338	0.863	0.751	0.879	0.906	0.907	0.932
20	0.844	0.513	0.894	0.779	0.881	0.913	0.898	0.939
21	0.830	0.390	0.854	0.770	0.892	0.895	0.916	0.949
22	0.827	0.424	0.873	0.752	0.860	0.914	0.877	0.911
23	0.800	0.281	0.848	0.737	0.864	0.904	0.884	0.913
24	0.837	0.366	0.880	0.760	0.878	0.909	0.904	0.941
25	0.809	0.363	0.870	0.736	0.864	0.922	0.876	0.885
26	0.841	0.422	0.886	0.784	0.890	0.918	0.917	0.931
27	0.830	0.509	0.865	0.770	0.893	0.911	0.908	0.942
28	0.791	0.377	0.870	0.719	0.866	0.910	0.883	0.898
29	0.851	0.421	0.873	0.803	0.885	0.912	0.906	0.946
30	0.848	0.447	0.888	0.788	0.873	0.911	0.893	0.937
31	0.800	0.463	0.876	0.723	0.856	0.920	0.873	0.885
32	0.821	0.468	0.876	0.753	0.847	0.905	0.870	0.909
33	0.821	0.490	0.883	0.746	0.885	0.905	0.910	0.923
34	0.846	0.478	0.879	0.780	0.893	0.906	0.913	0.940
35	0.816	0.402	0.865	0.754	0.873	0.903	0.888	0.912
36	0.826	0.327	0.847	0.765	0.864	0.894	0.883	0.922
37	0.813	0.289	0.851	0.754	0.873	0.910	0.880	0.892
38	0.818	0.430	0.861	0.759	0.882	0.902	0.911	0.941
39	0.845	0.415	0.874	0.798	0.878	0.903	0.903	0.943
40	0.820	0.476	0.853	0.751	0.881	0.902	0.900	0.922
41	0.806	0.386	0.872	0.709	0.844	0.913	0.854	0.900
42	0.847	0.405	0.863	0.784	0.883	0.913	0.890	0.936
43	0.810	0.483	0.889	0.712	0.851	0.913	0.884	0.897
44	0.847	0.460	0.884	0.781	0.885	0.920	0.901	0.932
45	0.825	0.325	0.848	0.746	0.877	0.913	0.883	0.901
46	0.831	0.333	0.869	0.759	0.878	0.912	0.890	0.908
47	0.850	0.452	0.873	0.795	0.882	0.908	0.898	0.946
48	0.847	0.428	0.887	0.775	0.895	0.911	0.913	0.941
49	0.831	0.498	0.894	0.753	0.861	0.919	0.907	0.940
50	0.834	0.399	0.872	0.756	0.873	0.908	0.886	0.920
Variance:	2.38E-04	5.10E-03	2.76E-04	5.59E-04	1.99E-04	5.38E-05	2.42E-04	4.58E-04
Std. dev.	0.01543	0.07143	0.01660	0.02365	0.01411	0.00734	0.01555	0.02139
Average:	0.82618	0.42194	0.87242	0.75491	0.87347	0.90964	0.89127	0.91828
Min:	0.79112	0.25900	0.83389	0.69712	0.84435	0.89420	0.85408	0.84225
Max:	0.85074	0.58100	0.91431	0.80256	0.89923	0.92399	0.91989	0.95141