

Kari Meling Johannessen

## Towards Improved Sheep Roundup

Using Deep Learning-Based Detection on Multi-Channel RGB and Infrared UAV Imagery

Master's thesis in Master of Science in Engineering and ICT

Supervisor: Hongchao Fan and Svein-Olaf Hvasshovd

June 2020



Kari Meling Johannessen

# **Towards Improved Sheep Roundup**

Using Deep Learning-Based Detection on Multi-Channel RGB and Infrared UAV Imagery

Master's thesis in Master of Science in Engineering and ICT  
Supervisor: Hongchao Fan and Svein-Olaf Hvasshovd  
June 2020

Norwegian University of Science and Technology  
Faculty of Engineering  
Department of Civil and Environmental Engineering



Norwegian University of  
Science and Technology



## Abstract

Each year, approximately 2.1 million sheep are released to graze freely in vast, forest-covered, and mountainous areas throughout Norway. At the end of the grazing season, farmers must find and round up their sheep. This can be a time consuming and challenging process because of the large and cluttered nature of the sheep grazing environment.

This thesis proposes a system for automatic sheep detection using UAV images to aid farmers in finding their sheep at the end of the grazing season. The goal is to propose and develop a deep learning model that automatically detects sheep in RGB and infrared UAV images and evaluate how well this model meets performance and processing speed requirements of a real-world application. Furthermore, the research questions compare performance of models that fuse RGB and infrared data with models using either RGB or infrared as input, and explore the impact of varying model complexity, fusion location, and input resolution on performance.

Based on a review of previous work on multi-channel image fusion networks and the current state of deep learning and object detection, a model architecture is designed to specifically address the task of automatic sheep detection in RGB and infrared images. Due to the low localisation quality requirement of the task, bounding box outputs are not required. Instead, the model head outputs a simple fixed size grid probability mask.

Several models were trained with a range of configurations to identify the set of optimal solutions for maximising average precision and minimising inference time. Results show that fusion of RGB and infrared data in a single model yields better average precision results than using data separately. The set of optimal solutions achieve average precision scores in the range of 69.9% to 96.3% with inference times ranging from 0.1 to 0.6 seconds per image. At a confidence threshold of 0.5, the most accurate network achieves a grid precision of 97.7% and a recall of 90.1%. This corresponds to the detection of 97.5% of the sheep in the validation dataset. The high-performance results achieved shows that automatic detection of sheep in multi-channel UAV images can be a great contribution towards improved sheep round up.

## Sammendrag

Hvert år slippes cirka 2,1 millioner sauer for å beite fritt i store, skogkledde og fjellrike områder i hele Norge. På slutten av beitesesongen må bøndene finne og samle inn sauene sine. Dette kan være en tidkrevende og utfordrende prosess på grunn av beite-områdetets store og uoversiktlige natur.

Denne masteroppgaven foreslår et system for automatisk gjenkjenning av sauer ved å bruke UAV-bilder for å hjelpe bønder med å finne sauene deres mot slutten av beitesesongen. Målet er å foreslå og utvikle en dyp læringsmodell som automatisk oppdager sauer i RGB og infrarøde UAV-bilder og evaluere hvor godt denne modellen oppfyller kravene til ytelse og prosesseringshastighet. Videre sammenlignes ytelsen til modeller som kombinerer RGB og infrarøde bilder med modeller som bruker enten RGB eller infrarøde bilder, og undersøker effekten av varierende modellkompleksitet, nettverkslokasjon for sammenslåing av RGB og infrarød data og bildeoppløsning.

Basert på en gjennomgang av tidligere arbeid med flerkanals bildefusjonsnettverk og den moderne dyp læringsteknologiens evne til objektgjenkjenning, er det utviklet en modellarkitektur spesifikt designet for å håndtere oppgaven med automatisk gjenkjenning av sauer i RGB og infrarøde bilder. På grunn av oppgavens lave kvalitetskrav til lokalisering, kreves det ikke at nettverket nødvendigvis skal finne hver enkelt sau. I stedet vil modellen predikere en enkel ruteformet sannsynlighetsmaske som sier noe om sannsynligheten for sau innenfor et gitt område.

Flere modeller ble trent med ulike konfigurasjoner for å identifisere settet med løsninger som gir optimal avveining mellom gjennomsnittlig presisjon og rask prosesseringsstid. Resultatene viser at fusjon av RGB og infrarød data i en modell gir bedre resultater enn å bruke disse dataene hver for seg. De beste modellene oppnår gjennomsnittlig presisjon på mellom 69,9% og 96,3% med prosesseringsstider mellom 0,1 og 0,6 sekunder per bilde. Med en konfidensterskel på 0,5, oppnår den mest nøyaktige modellen en presisjon på 97,7% og en tilbakekalling på 90,1%. Dvs. at av 97,5% av sauene i valideringsdatasettet ble identifisert. Dette viser at automatisk gjenkjenning av sau i flerkanals UAV-bilder har stort potensiale til en forbedret og mer effektiv saueinnsamling.

## Preface

This paper is a master thesis written for the Department of Civil and Transport Engineering at the Norwegian University of Science and Technology (NTNU) in Trondheim, Norway. It is part of the study program Engineering and ICT, with a specialisation in Geomatics.

Prior to beginning work on this thesis, I completed a specialisation project, which examined the use of deep learning-based object detection on detecting sheep in colour (RGB) images. This thesis expands and builds on research from this specialisation project. As such, there is some overlap between the previous work in the specialisation project and this thesis especially relating to background theory. Since it cannot be expected for the reader of this thesis to have read the specialisation project and in order to give a more seamless reading experience, some of the sections from the project preceding this thesis are included and adapted into this report. An overview of the sections that are adapted or included from the previous work are listed in appendix A.1. The thesis was written in the spring of 2020.

I would like to thank my supervisors, Hongchao Fan and Svein-Olaf Hvasshovd for your invaluable weekly encouragement and guidance during my work on this thesis. I would also like to thank my friends and family for insightful input and discussions. Finally, thank you Dirk for your feedback, love and support.

Trondheim, 10-06-2020

# Contents

<b>Abstract</b> . . . . .	<b>i</b>
<b>Sammendrag</b> . . . . .	<b>ii</b>
<b>Preface</b> . . . . .	<b>iii</b>
<b>Contents</b> . . . . .	<b>iv</b>
<b>List of Figures</b> . . . . .	<b>vi</b>
<b>List of Tables</b> . . . . .	<b>viii</b>
<b>1 Introduction</b> . . . . .	<b>1</b>
1.1 Background and Motivation . . . . .	1
1.2 Goals and Research Questions . . . . .	2
1.3 Research Method . . . . .	3
1.4 Scope and Limitations . . . . .	3
1.5 Thesis Structure . . . . .	3
<b>2 Background Theory and Related Work</b> . . . . .	<b>4</b>
2.1 Sheep Grazing and Roundup . . . . .	4
2.1.1 The grazing season . . . . .	4
2.1.2 Existing Technology . . . . .	4
2.2 Automatic Sheep Detection in UAV images . . . . .	6
2.3 Object Detection . . . . .	8
2.3.1 Object detection Metrics . . . . .	8
2.3.2 Detection datasets . . . . .	10
2.4 Deep Learning . . . . .	11
2.4.1 Convolutional Neural Network (CNN) Architectures for Computer Vision . . . . .	11
2.4.2 The Building Blocks of a CNN . . . . .	11
2.4.3 Influential Backbone Architectures . . . . .	16
2.4.4 Training a Neural Network . . . . .	18
2.4.5 Influential Architectures for Object Detection . . . . .	20
2.5 Multi-channel Image Fusion . . . . .	22
2.6 Camera Calibration . . . . .	23
2.6.1 Lens Distortion . . . . .	23
2.6.2 Affine Transformation . . . . .	24
2.7 Summary . . . . .	25
<b>3 Project Description</b> . . . . .	<b>27</b>
3.1 Envisioned Solution . . . . .	27
3.2 Requirements . . . . .	28
3.3 Sheep Dataset . . . . .	31
3.4 Hardware Constraints . . . . .	36
<b>4 Method</b> . . . . .	<b>37</b>
4.1 Data Preprocessing . . . . .	37
4.1.1 Data Selection and Sampling . . . . .	37
4.1.2 Camera Calibration and Image Alignment . . . . .	39
4.2 Software Environment . . . . .	42
4.3 Neural Network Architectural Design Choices . . . . .	42
4.3.1 Backbone Neural Network . . . . .	42
4.3.2 Network Head . . . . .	43
4.3.3 Fusion Network . . . . .	44
4.4 Training . . . . .	46
4.5 Experimental Variables . . . . .	50



4.5.1	Performance Metrics . . . . .	50
4.5.2	Independent Variables . . . . .	51
4.5.3	Control Variables . . . . .	52
4.6	Inference and Evaluation . . . . .	53
<b>5</b>	<b>Results</b> . . . . .	<b>54</b>
5.1	Data Preprocessing . . . . .	54
5.1.1	Data selection and sampling . . . . .	54
5.1.2	Image Alignment . . . . .	55
5.2	Experimental Results . . . . .	59
5.2.1	Model Configurations . . . . .	59
5.2.2	Best Model Performance . . . . .	62
5.2.3	Case Studies . . . . .	68
<b>6</b>	<b>Discussion</b> . . . . .	<b>71</b>
6.1	Data Preprocessing . . . . .	71
6.1.1	Data Sampling . . . . .	71
6.1.2	Image alignment . . . . .	71
6.2	Model Configuration Experiments . . . . .	73
6.3	Practical implications and Limitations . . . . .	75
<b>7</b>	<b>Conclusion and Future Work</b> . . . . .	<b>77</b>
7.1	Future Work . . . . .	77
	<b>Bibliography</b> . . . . .	<b>79</b>
<b>A</b>	<b>Appendix</b> . . . . .	<b>84</b>
A.1	Overview of Sections Adapted or Included from Specialisation Project . . . . .	85
A.2	Sample of Labelled Images In Training Dataset . . . . .	86
A.3	Sample of Images and Predictions on Validation Dataset . . . . .	87
A.4	Sample of Images and Predictions on T1 Datasets . . . . .	88
A.5	Sample of Images and Predictions on T2 Datasets . . . . .	89
A.6	Results of All Models . . . . .	90

## List of Figures

1	A modern adult sheep with electronic ear tag, bell and radio-bell. Image origin: [1] . . . . .	5
2	Example of opening by reconstruction, a traditional image processing method on infrared images. Image origin: [2] . . . . .	7
3	Object Detection to identify and locate sheep. . . . .	8
4	Intersection over Union (IoU). . . . .	9
5	Examples of true positive (TP), false positive (FP) and false negative (FN). . . . .	9
6	Trade-off between precision and recall. Image on the left has high precision but low recall. Image on the right has low precision but high recall. . . . .	10
7	An example of a CNN architecture used for image classification. Image modified from original: [3] . . . . .	11
8	Simple example of a convolution operation using a 4x4 image and a 3x3 filter. . . . .	13
9	Simple example of unpooling by nearest neighbour interpolation . . . . .	14
10	Simple visualisation of a transpose convolution operation using a 2x2 image and a 3x3 filter with a stride of 2 and output padding of 1. . . . .	15
11	Inception-v1 architecture. Figure origin: [4] . . . . .	16
12	ResNet-50 architecture. Figure origin: [4] . . . . .	17
13	ResNeXt-50 architecture. Figure origin: [4] . . . . .	18
14	An example of underfitting, overfitting and fitting a model just right. Image origin: [5] . . . . .	18
15	Extending CNN model to accept multiple input channels. Image adapted from [6] . . . . .	22
16	Fusion Network - a method of taking advantage of multiple input channels whilst keeping the advantages of transfer learning. The two input images are first run through two parallel subnetworks who's outputs are later fused somewhere midways in the network. Image origin: [7] . . . . .	23
17	Envisioned system to find sheep. . . . .	27
18	Sample search areas around Storlidalen in Oppdal . . . . .	28
19	Some examples of sheep races that exist in Norway, [8]. . . . .	30
20	Definition of extra small (xS), small, medium, large and extra large (xL) sheep. Boxes show the maximum size for the sheep in each classification in relation to the full image dimension of 3040 x 4056. Images are classified as one of the above size categories by the median diagonal length (D) of all adult sheep bounding boxes. Using the fact that the average adult sheep is 1.3m long, [9], ground sampling distance (GSD) and flight height (H) are estimated for each category by equations 3.1 and 3.2 . . . . .	32
21	A sample of the sheep in the dataset grouped by colour. The exact transition between white, grey and black is somewhat fuzzy. . . . .	33
22	Sample of other animals and humans present in the dataset. . . . .	34
23	Sample of labelled image pairs in the sheep dataset. Labelled RGB image is shown on the left and corresponding infrared image is shown on the right. . . . .	35
24	DJI Mavic 2 Enterprise Dual (M2ED). . . . .	36
25	An overview of the processes performed on the data to prepare, build, utilise and evaluate the sheep detection models. . . . .	37
26	MSX infrared images are excluded from use . . . . .	38
27	Images with too small or too large sheep are excluded from use. Accepted flight heights correspond to an approximate range of 14 to 85 meters. . . . .	38
28	Three step process for aligning the infrared and RGB images . . . . .	40
29	Setup for camera calibration showing the overhead projector, the dual imaging UAV and a sample of the RGB and infrared image pairs taken by the camera. . . . .	41

30	Sub-sample of the in total 72 checker-board photos with marked corner points used for camera calibration . . . . .	41
31	Images of torches, visible in both visible and infrared images . . . . .	42
32	Network architecture used for RGB only or infrared only input . . . . .	43
33	Desired grid result when an infrared image is processed by model. Red boxes are bounding boxes that have been transformed from the RGB coordinate system and green-shaded grid cell represent cells where the detector has made a positive sheep detection. As shown in the image, the transformed bounding boxes do not align well with the sheep in the image since the sheep are moving in their environment, however the grid classifications are still mostly correct . . . . .	44
34	Output of YOLO v1 on the pascal VOC dataset compared to the conceptual grid output of the sheep dataset. . . . .	44
35	Fusion Network. This architecture is used for accepting both RGB and infrared input . . . .	45
36	The various depths of ResNet where fusion is attempted. ResNet layer names and detailed architectures are described in Table 2. Fusion at depths 3,4 and 5 is done by concatenation and $1 \times 1$ convolutions, whereas fusion at depth 6 is done with concatenation followed by an extra fully connected layer . . . . .	45
37	Fusion layer architecture. Fusion is done by concatenation followed by either $1 \times 1$ convolutions if fusion is performed mid-ways in the network or an extra fully connected layer if fusion is performed after the fully connected layer. . . . .	46
38	The Training Process . . . . .	46
39	Training Augmentation Pipeline . . . . .	47
40	How ground truth grids are defined from bounding boxes . . . . .	49
41	The processes considered by inference time . . . . .	51
42	RGB image points and reprojected infrared points . . . . .	56
43	Plots of reprojection error vs image number and distance from the image centre . . . . .	57
44	Result of Image alignment shown by overlaying edge features of the RGB image onto a transformed infrared image . . . . .	57
45	Boxes from the RGB coordinate system transformed to the infrared coordinate system. Alignment appears to be good in cases when sheep are standing still but not as good in the case when the sheep are moving. . . . .	58
46	Validation average precision performance of all models against inference time grouped by input type. Some points of interest are labelled. Eg. 'r18_f4_rgb1024' is a model trained with a ResNet18 backbone, fuse level 4 and RGB crop resize shape of 1024. . . . .	60
47	Validation average precision and inference time grouped by model backbone and input type. Numerical values for this graph can be found in Table 13. . . . .	61
48	The effect of RGB resolution on validation average precision and inference time performance for the RGB and fusion (RGB+I) models with backbone model ResNet18. RGB input size is the size that the raw $1200 \times 1200$ -pixel RGB image crop is resized to prior to being processed by the model. . . . .	62
49	Precision $\times$ Recall curve for the ResNeXt50 I, RGB and RGB+I models. The red shows which precision and recall values that are obtained when selecting a threshold of 0.5. . . . .	64
50	Examples of predictions made on the validation dataset by the ResNeXt50 I, RGB and RGB+I models. . . . .	65
51	Average Precision grouped by dataset and median adult sheep size for RGB + I model . . .	66
52	Sheep Recall by colour. . . . .	67
53	Average precision performance of the best Infrared, RGB and Fusion models on the Validation, T1 and T2 datasets . . . . .	68
54	The difference in sheep recall for lamb vs. adult sheep . . . . .	69
55	How well the model distinguished other animals in the dataset from sheep . . . . .	70
56	The size of the reprojection error relative to the size of the image . . . . .	72

## List of Tables

1	Overview of existing radio bell tracking products. A checkmark indicates that the feature is offered by the product . . . . .	6
2	ResNet architectures. Downsampling is performed by conv3 1, conv4 1, and conv5 1 with a stride of 2. Table origin: [10] . . . . .	17
3	Object Detection Requirements. . . . .	30
4	The number of images in the sheep dataset grouped by month, distribution of sheep size, distribution of free ranging versus fenced sheep and distribution of MSX versus normal infrared format. In addition, the sheep races represented in the dataset are marked. * Sheep races are: <b>1.</b> Norwegian White Sheep, <b>2.</b> Norwegian Pelssau and <b>3.</b> Old Norwegian Spælsau. . . . .	31
5	Number of sheep in the dataset grouped by month, colour and life stage. . . . .	32
6	DJI Mavic 2 Enterprise Dual (M2ED) specifications. . . . .	36
7	Explanation of Chosen Augmentations . . . . .	48
8	The learning rate and batch size used for experiments . . . . .	52
9	The number of images in the sheep dataset after sampling. Images are grouped by distribution of sheep size and distribution of free ranging versus fenced sheep. In addition, the sheep races represented in the dataset are marked. * Sheep races are: <b>1.</b> Norwegian White Sheep, <b>2.</b> Norwegian Pelssau and <b>3.</b> Old Norwegian Spælsau. . . . .	55
10	The number of sheep in the dataset after sampling grouped by sheep colour and sheep life stage. . . . .	55
11	The calculated distortion coefficients . . . . .	55
12	Average precision, precision, recall and inference time results for models on the pareto front. The best values are highlighted in bold. Grid precision, grid recall and sheep recall are for the validation dataset at a confidence threshold of 0.5 . . . . .	60
13	Result of models grouped by model backbone and input type. All models shown in this comparison are trained on RGB crop resize shape of 1024, infrared crop resize shape of 64 and fused at depth 4 (for RGB+I). An average is taken in the cases where more than one model with the same configuration exists. The best values are highlighted in bold. . . . .	61
14	Average precision, precision and recall for ResNeXt50 models. Average precision is grouped by median adult sheep size per image as defined in Figure 20. Grid precision and recall is calculated with using a confidence threshold of 0.5. . . . .	63
15	Sheep recall grouped by input type, sheep colour and sheep life stage. . . . .	67
16	Results recorded for all trained models. . . . .	90

# 1 Introduction

This chapter gives the background and motivation behind the thesis as well as the main goal and research questions. In addition, an outline of the research methods used to reach the goal and to investigate research questions are presented. Finally, the scope and main limitations of the thesis are presented.

## 1.1 Background and Motivation

Each year, approximately 2.1 million sheep are released to graze freely in vast, forest-covered and mountainous areas throughout Norway, [11]. The sheep's ability to live on vegetation that is naturally found in their free grazing environments make them an efficient source of food and wool.

However, the process of rounding up all the sheep at the end of the grazing season is a challenging and time-consuming task for many sheep farmers. The grazing area is large and cluttered, and the sheep do not always wander in predictable areas. Although technologies that help farmers find their sheep exist, [12, 13, 14, 15], these are not perfect. Radio-bells, a commonly used GPS tracking technology for sheep, are limited by a high unit price and the need for data communication in areas with typically poor mobile coverage. The use of unmanned aerial vehicles (UAVs), commonly known as drones, can also be used to survey areas of challenging terrain to find sheep. However, searching for sheep in UAV images is currently done manually, which can be difficult, boring and error prone.

With the current state of the art (SoTA) deep learning-based object detection algorithms, [16, 17, 18, 19] being able to achieve impressive results on large, challenging datasets such as COCO, [20], there is reason to believe that a deep-learning based approach could be applied to UAV images to detect sheep.

This thesis proposes a system for automatic detection of sheep in UAV images to aid sheep farmers in collecting their sheep at the end of the grazing season. Previous research on this topic, [21, 22, 23] has found that deep learning-based methods can be applied to this task but the success of a deep learning-based approach is dependent on a large amount of training data. Moreover, all previous attempts involving the use of either visible light (RGB) image or infrared images suggest that these two image types contain valuable information that collectively can be used to improve sheep detection. Research in this thesis builds upon previous work by exploring a larger and more realistic dataset as well as considering both RGB and infrared input in the solution.

## 1.2 Goals and Research Questions

**Goal** *Propose and develop a deep learning model that automatically detects and locates sheep in RGB and infrared UAV images and evaluate the extent to which the model meets performance and processing speed requirements of a real-world application.*

The main goal of this thesis is to propose and develop a deep learning model for RGB and infrared drone images that can aid farmers in finding their sheep at the end of the grazing season. For the model to be applicable in a real-world application, the model predictions should be precise, find most of the targets and have a reasonable processing time. As a result, two performance metrics are assessed. These are average precision and inference time.

This goal is further divided into three research questions, which address the main factors that affect detection performance and inference time of the deep learning-based sheep detection model. By experimenting on varying these factors, an extensive trade-off analysis between performance and inference time can be done.

**RQ1** *How does the input data type affect detection performance and inference time of a deep learning-based sheep detection model?*

Input data type refers to the image type analysed by the model. This can either be RGB only, infrared only or a combination of both RGB and infrared.

**RQ2** *How does the network design affect detection performance and inference time of a deep learning-based sheep detection model?*

The network design refers network design parameters such as the number of layers, convolutional filter sizes etc. Current state of the art backbone networks ResNet and ResNeXt are tested. Furthermore, a fusion network that fuses two parallel sub networks is designed in order to accept both RGB and infrared input. Experiments are performed to determine the best location in the network to perform fusion.

**RQ3** *How does the data resolution affect detection performance and inference time of a deep learning-based sheep detection model?*

Data resolution refers to the degree of down sampling of the input data prior to it being passed to the model. A higher resolution will likely improve detection performance; however the question is whether the slower inference time due to the higher resolution is an acceptable trade-off for improved performance.

## 1.3 Research Method

The steps taken to realise the goals mentioned in the previous section involve the following:

1. A literature review to understand the state of current technology related to deep learning, fusion of multi-channel imagery in deep learning applications and image alignment. Findings of the literature review build a basis on which to design, train and evaluate the sheep detection models proposed in the thesis.
2. Identification of the requirements and constraints to the sheep detection model.
3. Preprocessing data by selecting a suitable sample and performing camera calibration and affine transformations to align the RGB and infrared image pairs.
4. Designing a convolutional neural network (CNN) architecture that is hand crafted to the multi-channel input and low localisation quality requirement of the sheep detection task.
5. Training several models with a range of configurations to provide basis for answering the research questions and discovering the set of optimal solutions with regards to average precision and inference time.

## 1.4 Scope and Limitations

The scope of the thesis is analysing the image processing part of the sheep detection system. The thesis does not address data collection and transfer between devices or how the ultimate user interface of such a system would look.

The validity of the results is limited by a relatively small dataset captured in a limited geographical area of Norway so performance results will not necessarily generalise as well to other areas of Norway. Moreover, reported results are dependent on the hardware that was available.

Time is also a limitation. Training on the available hardware takes several hours and often more than a day. It is therefore not feasible to test and repeat every combination of model configurations.

## 1.5 Thesis Structure

**Chapter 1: Introduction** introduces the background and motivation behind the thesis as well as the main goal and research questions. In addition, an outline of the research method and limitations is presented.

**Chapter 2: Theory and Related Work** presents the relevant fundamental theory and related work on which the thesis is based. This includes theory about sheep grazing, object detection, deep learning, multi-channel image fusion and camera calibration.

**Chapter 3: Project Description** presents an overview of the envisioned solution, the desired requirements to the final performance of the object detector. In addition, the constraints in form of available data and hardware are presented.

**Chapter 4: Method** explains the decisions made and steps taken to create, train and evaluate the sheep detection models.

**Chapter 5: Results** presents the results obtained from following the approach outlined in the previous chapter.

**Chapter 6: Discussion** critically discusses the validity and implications of the results presented in the previous chapter. The findings are evaluated with regards to each of the research questions

**Chapter 7: Conclusion and Future Work** reviews the key points of the thesis and explains its relevance. In addition, recommended future work based on the findings of the thesis is drafted.

## 2 Background Theory and Related Work

This chapter presents the relevant fundamental theory and related work on which the thesis is based. This includes theory about sheep grazing, object detection, deep learning, multi-channel image fusion and camera calibration.

### 2.1 Sheep Grazing and Roundup

This section gives insight into the current state of the sheep grazing process and highlights which technologies currently exist. The Norwegian agricultural cooperative Nortura, [24], recently released a thematic report, which gives information and recommendations on how to manage sheep during the grazing season. NTNU professor S-O. Hvasshovd, who has been researching modernisation of raising sheep for several years and actively carrying out research by participation, has also given insight into the processes surrounding sheep grazing and roundup.

#### 2.1.1 The grazing season

Every summer, approximately 2.1 million sheep are released to graze freely on large pasture areas in Norway, [11]. On average, 40% of a sheep farmers feeding source comes from here. Outback sheep grazing also has the advantage that it prevents nature from overgrowing, which is important for maintaining the current biological diversity, [25].

During the grazing season, the farmers are required by law to inspect their sheep at least once a week and to keep documentation of these inspections, [25]. For this purpose, it is common that farmers form teams and share the inspection responsibilities. Since sheep do not always spread ideally over the grazing areas, injury and loss of oversight may occur. In dealing with this problem, some farmers hang posters to encourage people to share information about injured or lost sheep. Fences and strategically placed salt blocks may also help encourage sheep to graze in desired locations.

When lambs have reached a desired weight or when outdoor food resources are reduced in quality due to seasonal changes, it is time to collect the sheep. This is often a large event that requires a lot of manpower and time. Cooperation between farmers and help from family and neighbours is therefore crucial for a successful sheep collection. S-O. Hvasshovd describes the typical sheep round up as happening in three phases, [26]:

1. **Main roundup 1:** The first roundup phase often involves help from many people and sheep dogs and commonly runs over 1-2 weekends. In this time, approximately 90% of sheep are collected.
2. **Main roundup 2:** The farmer goes over the same area as in the first round up phase to collect the missed sheep. This phase can last a couple of weeks and the farmer has much less help. In this second round up 5% to 10% of sheep are collected.
3. **Collecting Stragglers:** Typically, 10 to 20 sheep spread across 5-6 groups are left uncollected by this stage. These sheep have often wandered outside the typical grazing terrain and are especially hard to find. As a result, this phase can be very time consuming and a source of frustration for the farmer.

#### 2.1.2 Existing Technology

There exist some technical solutions on the market today that farmers can use to keep track of their sheep. These are bells, radio bells, electronic ear tags and UAVs.





**Figure 1:** A modern adult sheep with electronic ear tag, bell and radio-bell. Image origin: [1]

## Bells

The simplest and cheapest technology that is commonly used to keep track of sheep is the bell. The bell is worn around the sheep's neck and will make noise when the sheep moves. As a result, the farmer can find nearby sheep by listening to the sound of bells, [1].

Bells are a practical and cheap way to help the farmer find his sheep, however it also has some downsides. For one, the bell can only be heard if the person searching is already in proximity of the sheep. Secondly, little research has been done to investigate the effect of the constant bell sound on the animal's well-being and whether this constant loud ringing can have a damaging effect on the animal's hearing. Moreover, it is also unknown whether the bells attract or repel predators, [1]. Finally, it is not possible to put bells on lamb as they will grow a lot during the grazing season.

## Radio Bells

Radio Bells are commonly used to track sheep while they are grazing. Some examples of these products are *Smartbjella*, [12], *Findmy*, [13] and *Telespor*, [14]. A summary of the price and features of these products are given in Table 1. These products work by having the sheep wear a GPS and radio tracking collar that transmits their location. Using this, sheep farmers can get an overview of their sheep's current and historical locations via a website or application. Other features offered by these products include geofencing, movement alarm and stress warning.

Radio bell tracking of sheep can be very advantageous to farmers as long as they are within an area that has mobile coverage. However, as shown in Table 1, the price per unit is quite high. Due to the high price, a common way to use these products is to only track a portion of the total sheep herd and assume that the sheep will stay in groups. Another limitation with this tracker is that it cannot be used on lam since lam are growing fast and therefore cannot wear traditional collars.

The communication technology used by *Smartbjella* and *Telespor* is Narrow-band Internet of Things (NB-IoT), which the providers claim covers larger areas than regular mobile internet such as 4G. The network is currently being expanded in Norway but there are still large areas that are missing coverage. *Findmy* uses low orbit satellite technology for communication and as a result is not dependent on internet coverage. This technology will work as long as the sheep are outdoors with a free view to the sky.

Some of the features offered by the radio bell products are:

- **Geofencing:** Geofencing works by having the user draw a virtual fence on a map and if one of the

tracked sheep enters or leaves this area then the farmer is notified and/or the sheep receives a light electric shock to prevent them from leaving the assigned area.

- **Movement Alarm:** Movement alarm will send a notification if a tracked sheep has been immobile over a longer period of time, typically 48 hours. This can be useful in order to detect sheep that are injured, stuck or dead.
- **Stress Warning:** Stress warning is a feature only offered by *Findmy*. The product analyses movement patterns of the sheep and is able to detect stressful movement behaviour in a herd. This can be useful as it will give an indication to the farmer that something is scaring the sheep and as a result, extraordinary measures can be taken to protect the sheep.

	Smartbjella	Telespor	Findmy
<b>Price per tracker (NOK)</b>	990	1124	1590
<b>Seasonal Subscription cost (NOK)</b>	99	124	229
<b>Battery life (years)</b>	10	1	3
<b>Tracking Technology</b>	NB-IoT,GPS	LTE-M, NB-IoT ,GPS	low orbit satellite, GPS
<b>GPS tracking</b>	✓	✓	✓
<b>Geofencing</b>	✓	✓	✓
<b>Movement alarm</b>	✓	✓	✓
<b>Stress warning</b>			✓

**Table 1:** Overview of existing radio bell tracking products. A checkmark indicates that the feature is offered by the product

### Electronic Ear Tags

Another common aid that can be used to track sheep is an electronic ear tag. By placing readers of these ear tags at a strategical location such as a salt block, farmers can get information about which sheep have 'checked in' to these locations. These are very affordable, costing under NOK 20 per unit. In addition to the tags, it is also necessary to have a radio at the location to transmit the information to the farmer.

### UAVs

In an article by 'Norsk Sau og Gjeit' (NSG) [15], UAVs are pointed out as being a cheap and effective technology for observing sheep during the grazing season. UAVs can easily fly over and capture images of challenging terrain. This can save the farmers valuable time and energy. However, currently the process of searching for sheep in UAV images is manual, which can be time consuming, boring and error prone.

### Summary of Existing Technologies

Radio Bells are a useful technology to help farmers locate their sheep, however as previously discussed their use is limited by mobile coverage requirements or expensive satellite tracking as well as a general high unit price. UAVs are also a useful technology being employed today, however detection of sheep in UAV images is not yet automated. As a result, an alternative solution such as the one suggested by this thesis could potentially be useful in helping the sheep farmers to find the last of their sheep. A description of the solution envisioned is presented in section 3.1.

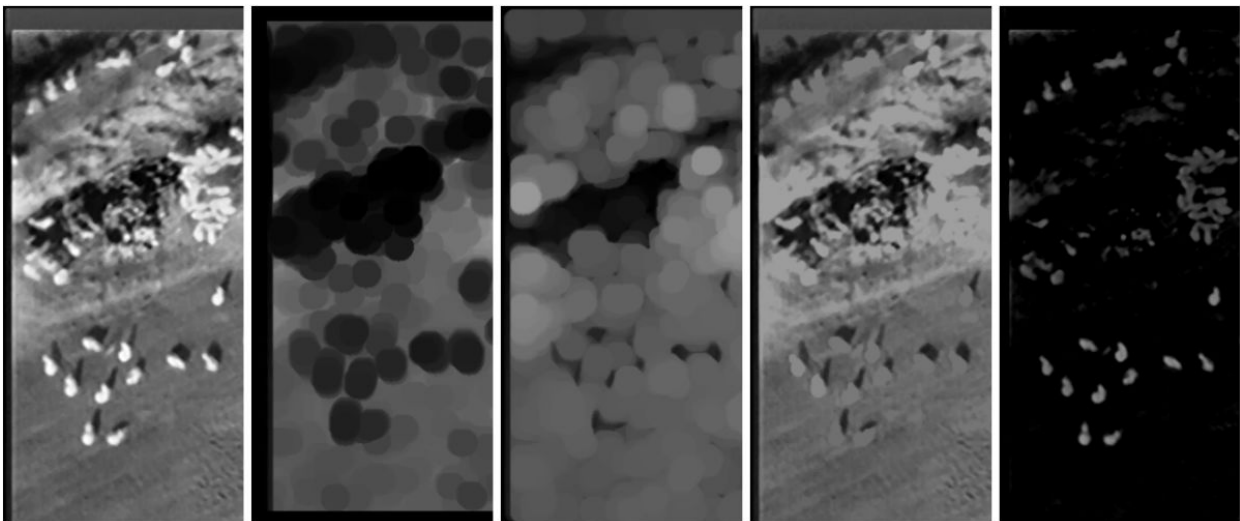
## 2.2 Automatic Sheep Detection in UAV images

Over the past two years, there have been some master theses and projects that explore the task of automatic sheep detection in UAV images, [2, 21, 22, 23]. This research explores either the use of infrared images

alone, [2] or RGB images alone, [21, 22, 23]. Infrared imagery has the advantage over conventional RGB images as they can distinguish animals based on body heat and can also detect animals at night, [27], however infrared cameras are more expensive and often lower resolution. On the other hand, RGB camera drones are common, affordable and the images contain useful information regarding colour and texture that cannot be found in infrared images. As a result, all the aforementioned research agrees that both RGB and infrared image channels contain valuable information and that combining information from both channels has potential to improve overall detection results.

The methods previously attempted range from traditional computer vision methods to more modern machine learning-based methods. Traditional approaches involve applying a series of operations such as filters, thresholding, erosion and dilation etc. to find hand crafted features of sheep. In comparison, a machine learning approach trains end to end, which means there is no need to hand craft features and the network automatically learns the most important features of the images, [28]. Deep learning approaches top the state of the art (SoTA) charts, however they are dependent on a large amount of data and computing power. If the amount of data is insufficient and the relevant features are simple and concrete, traditional methods may be superior, [28].

Rognlien, [2] used a traditional computer vision approach on infrared UAV images. Due to significant noise in the dataset, the data used was limited to 25 images. The authors found that the amount of data was insufficient to use machine learning but had some promising results using more traditional image processing methods. Using a traditional computer vision approach, the authors were able to detect 83.3% of the sheep with 85.7% precision, which is an acceptable result considering the very small amount of data.



**Figure 2:** Example of opening by reconstruction, a traditional image processing method on infrared images. Image origin: [2]

Ytterland and Winsnes, [21] used a UAV with a mounted RGB camera to capture images of sheep. The authors attempted to use various traditional computer vision methods such as applying filters and thresholding to the images to detect sheep. This worked relatively well for white sheep but was unsuccessful for non-white sheep. The solutions also had high rate of false positives. The authors suggest that combining results with infrared images would be useful to eliminate many of the false positive detections.

Muribø, [22] used the deep learning-based object detection architecture YOLOv3 to detect sheep in RGB images. YOLOv3 was chosen due to it having a good trade-off between performance and inference time. The author was able to achieve very good results, reporting recall and precision results of 94% and 99% respectively. However, the validity and applicability of the results are limited by the fact that all the images are captured in the same location and all consist of the same group of fenced sheep, grazing on a grassy field. This means that the test dataset is not independent, and the data is not representative of a real-world use case, which would contain many other objects as well as partially occluded sheep. Nonetheless, the promising results are a good proof of concept for a deep learning-based approach. The author suggests that improvements can be made by collecting more varied data, combining RGB images with infrared images

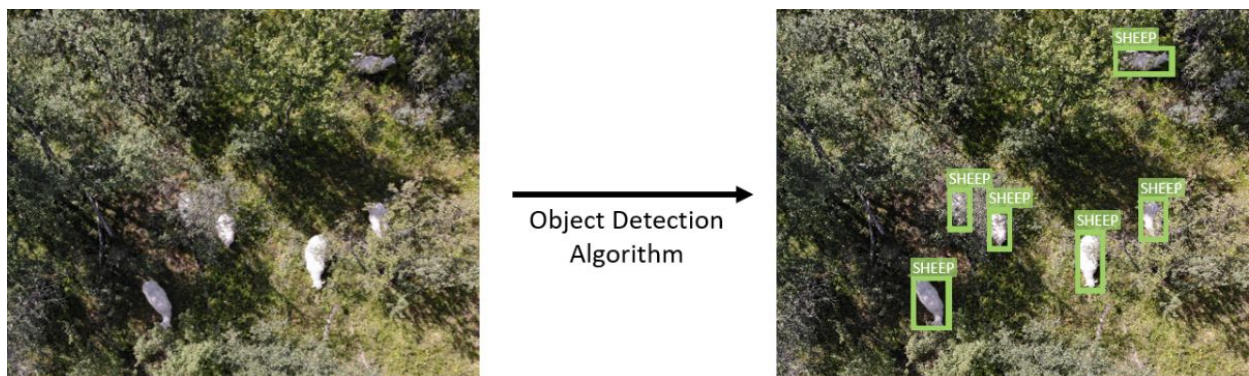
and including altitude information in order to eliminate proposals that are too large or too small.

In the project preceding this thesis, [23], the use of the deep learning multistage detector Libra-RCNN was tested on a collection of RGB images of sheep. An 86% recall with 91% precision was achieved using this method. These results are worse than [22], however this is expected because the dataset included much more realistic variation in scenes including free ranging sheep in challenging terrain and an independent validation dataset. As with the other projects looking at RGB only images, the project preceding this thesis also found white sheep easier to detect than other sheep colours. Despite a larger, more varied dataset, it was found that an even larger dataset would be beneficial to improve performance and validity of the model. In addition, [23] did not take processing time into consideration and reported an inference time of 36s per image, which is very impractical.

In conclusion, previous attempts at automatically detecting sheep in UAV images all agree on two things. Firstly, a machine learning-based approach has potential but requires a large and varied dataset in order to be successful. Secondly, infrared and RGB images both contain valuable information that could aid in the detection process and the two data types can complement each other well if combined. Until now, previous attempts have been limited to a very small dataset and to either one of RGB or infrared images. This thesis builds on this by attempting to combine the two image types and using a larger and more varied dataset than previous attempts.

## 2.3 Object Detection

Object detection is the task that is that this thesis addresses. Object detection is the process of locating and classifying objects in an image, [29]. Figure 3 shows an example of the desired result of applying an object detection algorithm on an image to detect sheep. An object detection algorithm takes an image as input and outputs bounding boxes for all the instances found in the image. For each output bounding box, the detector will also predict which class the object belongs to. In other words, object detection algorithms answer the question: *What objects and where?*, [30]. Object detection has a wide range of use cases. Facial detection, self-driving cars and visual search engines are just some examples of where object detection is being used, [31]. In this section, the definition of object detection and performance metrics used to evaluate object detection algorithms are discussed. Influential deep learning-based architectures for object detection are presented later in section 2.4.



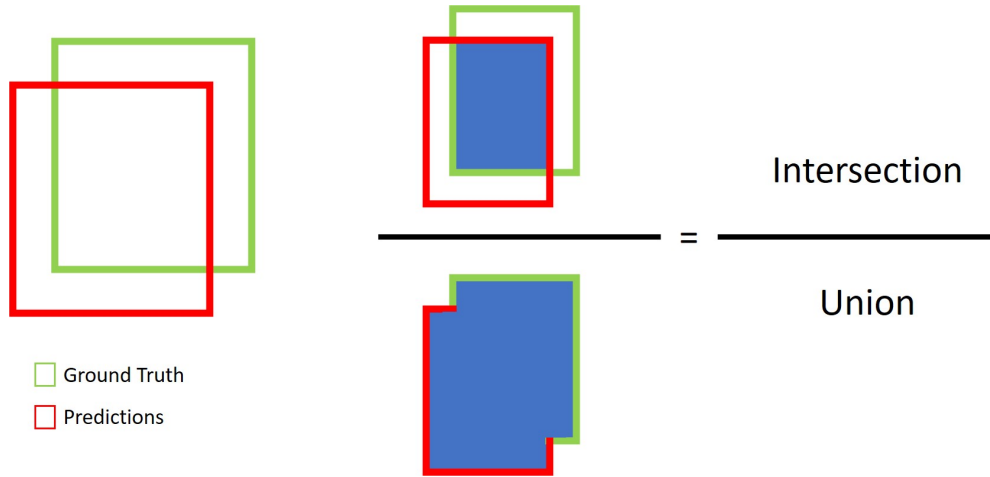
**Figure 3:** *Object Detection to identify and locate sheep.*

### 2.3.1 Object detection Metrics

In order to evaluate and compare the performance of object detection algorithms, it is necessary to use a common metric. Average Precision is most commonly used and is defined as the average precision over a set of evenly spaced recall values, [30]. Average precision is a good metric because it takes both precision and recall into account.

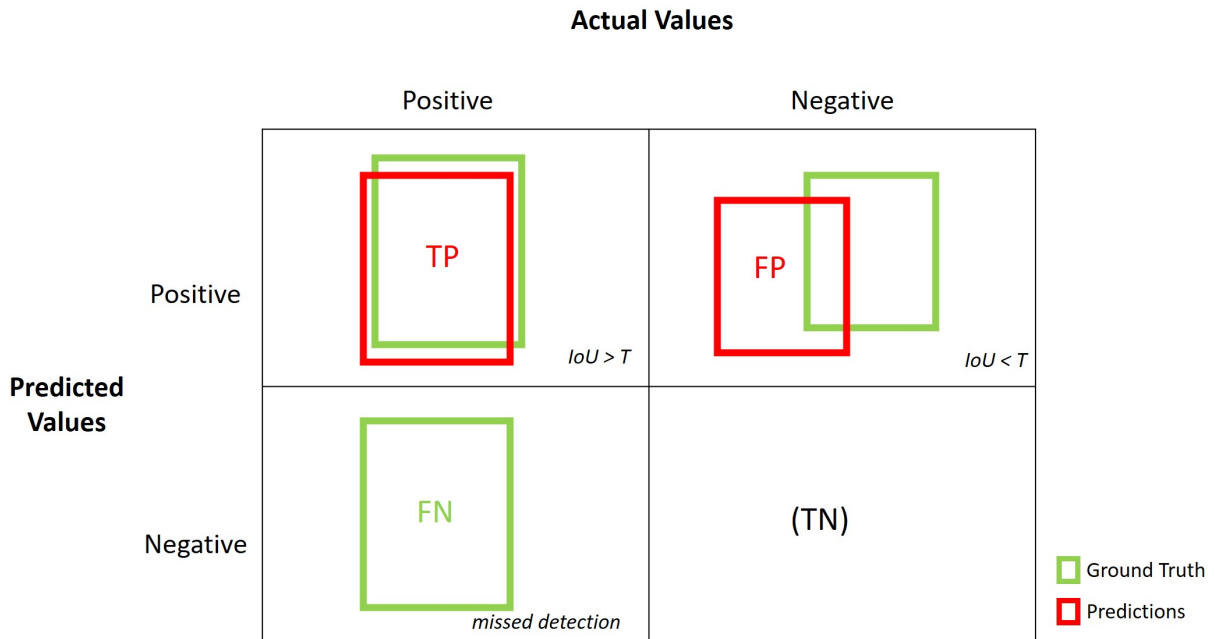
The quality of a predicted bounding box is judged by its degree of overlap with a given ground truth bounding box. This is called intersection over union (IoU) and is simply the intersection area divided by the

union area as shown in Figure 4.



**Figure 4:** Intersection over Union (IoU).

The IoU can be used to classify each predicted bounding box as either a true positive (TP) or a false positive (FP). A predicted bounding box is a TP if it has an IoU of more than a given threshold value with a ground truth box and it is a FP if it has an IoU of less than the threshold. A false negative (FN) occurs if the detector fails to detect a ground truth bounding box. Figure 5 shows some examples of occurrences of TPs, FPs and FNs. True negatives (TN) is the case where the detector correctly did not predict a bounding box. This is not considered because it is impossible to quantify.



**Figure 5:** Examples of true positive (TP), false positive (FP) and false negative (FN).

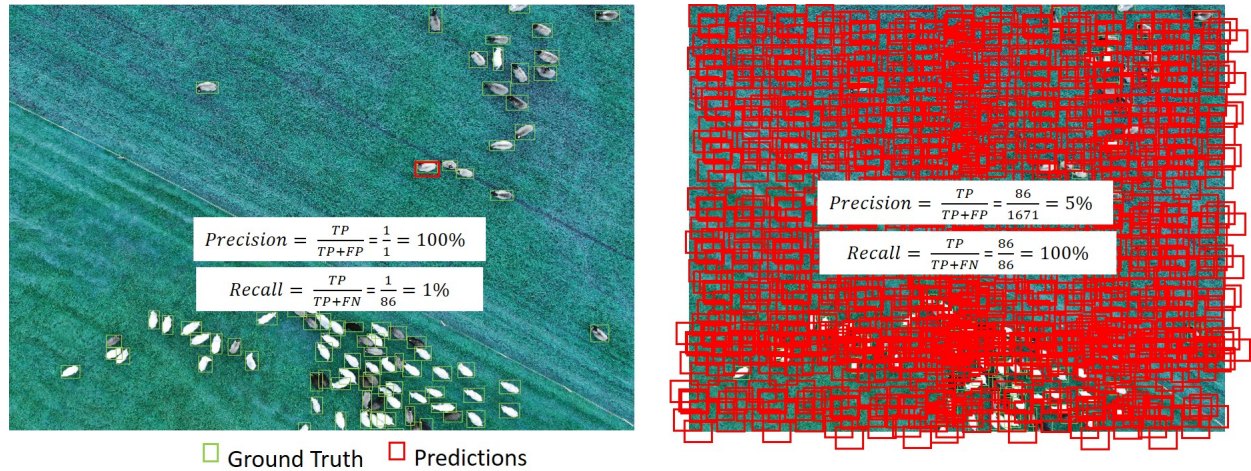
Precision is expressed by equation 2.1 and reveals the proportion of predictions that were correct. However, a weakness with precision as a detection metric is that it does not take into account all the detections that were missed by the object detector (FNs).

$$Precision = \frac{TP}{TP + FP} = \frac{\text{Correct predictions}}{\text{All predictions}} \quad (2.1)$$

In comparison, recall does take FNs into account and is expressed in equation 2.2. Recall reveals the proportion of all ground truth objects that the detector was able to detect. In order to improve recall, one could simply increase the number of predictions since this would increase the chance of detecting an object.

However, making more predictions would likely increase the number of FPs, which would be detrimental to the precision value. This trade-off between precision and recall is clearly demonstrated in Figure 6.

$$Recall = \frac{TP}{TP + FN} = \frac{\text{Correct predictions}}{\text{All ground truths}} \quad (2.2)$$



**Figure 6:** Trade-off between precision and recall. Image on the left has high precision but low recall. Image on the right has low precision but high recall.

A good object detector should perform well on both precision and recall. As a result, average precision is a good metric as it is average precision values over a range of recall values. A common IoU threshold value to use is 0.5. When the IoU threshold is set to 0.5, the average precision is referred to as  $AP_{50}$ . Recently, it is also common to also consider  $AP_{50:95}$ , which is the average average precision over IoU thresholds ranging from 0.5 (course localisation) to 0.95 (perfect localisation). Either metric is suitable, depending on how accurate the bounding boxes are required to be. If an object detector detects multiple classes, then a common metric used to describe detection is mean average precision (mAP), which is simply the average average precision of all the classes.

### 2.3.2 Detection datasets

There are three datasets that are commonly used to benchmark object detection results. These are Pascal VOC2007, Pascal VOC2012, [32] and Microsoft COCO (MSCOCO), [20].

#### VOC2012

The VOC2012 dataset has 20 categories and images are divided into training, validation and test splits with 2501, 2510 and 5011 images respectively.  $AP_{50}$  is used as the detection metric.

#### VOC2017

The VOC2017 dataset has 20 categories and as with VOC2012, the images are divided into training, validation and test splits with 5717, 5823 and 10991 images respectively. Annotations for the test dataset are not publicly available.  $AP_{50}$  is used as the detection metric.

#### Microsoft COCO

The COCO dataset is a large dataset with 80 object categories. The dataset consists of 118287, 5000 and 40670 labelled images in the training, validation and test sets respectively. Annotations for the test dataset are not publicly available. The coco detection challenge uses a range of average precision variants as detection metrics:

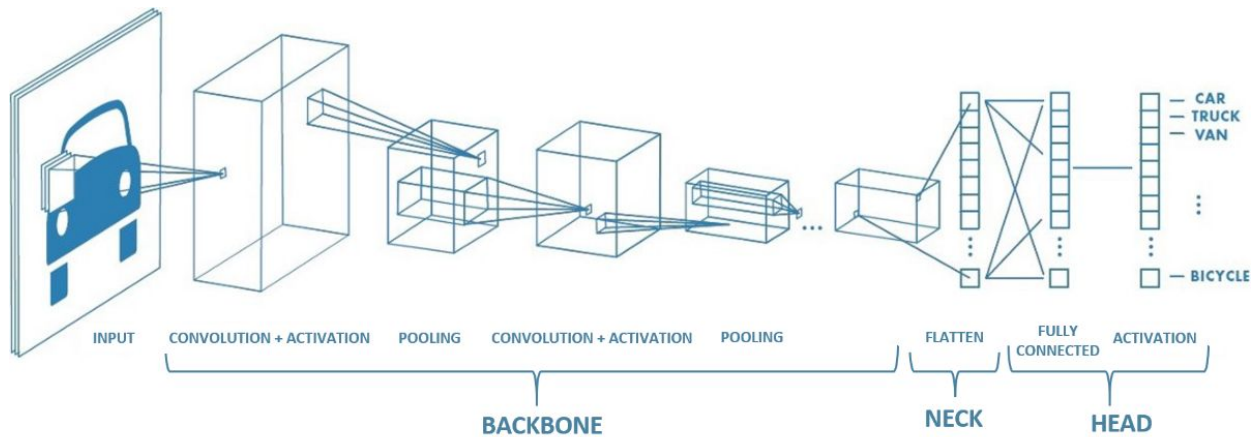
- $AP_{50:95}$ : mAP averaged over ten IoU thresholds: 0.5 : 0.95
- $AP_{50}$ : mAP at 0.50 IoU threshold
- $AP_{75}$ : mAP at 0.75 IoU threshold
- $AP_S$ :  $AP_{50:95}$  for small objects of area smaller than  $32^2$
- $AP_M$ :  $AP_{50:95}$  for objects of area between  $32^2$  and  $96^2$
- $AP_L$ :  $AP_{50:95}$  for objects larger than  $96^2$

## 2.4 Deep Learning

### 2.4.1 Convolutional Neural Network (CNN) Architectures for Computer Vision

Computer vision is a field of study that aims to automate tasks that the human visual system is able to perform, [33]. Examples of computer vision tasks are image classification, object detection, pose estimation, image segmentation and image captioning. Computer vision can be applied to 2D images, 3D images and video.

Modern computer vision primarily builds around one particular concept - convolutional neural networks (CNNs). A CNN is a form of deep learning that is effective at learning robust, high level features of an image due to its ability to exploit spatial/temporal relations, [34]. Figure 7 shows an example of a simple CNN architecture used for image classification.



**Figure 7:** An example of a CNN architecture used for image classification. Image modified from original: [3]

### 2.4.2 The Building Blocks of a CNN

As shown in Figure 7, a typical CNN architecture for computer vision can be said to contain three main building blocks, [35]:

1. **Backbone:** The backbone neural network is responsible for converting the input image into a feature map by applying a set of sequential convolutional filters, activation functions and pooling on the data.
2. **Neck:** The neck is responsible for connecting the backbone network to the head/heads. An example of a neck is a feature pyramid network, which is commonly used in CNN architectures for object detection.
3. **Head:** The head is responsible for computing the desired output from the feature map. An example of a network head is a set of fully connected layers that outputs class probability scores as shown in Figure 7. A CNN architecture can consist of multiple heads. Mask RCNN, [36] is a typical example of a network that has more than one parallel head. One head predicts bounding boxes whilst the other predicts object masks.

These three main building blocks are composed of some fundamental building blocks. As exemplified in Figure 7, a CNN backbone consists of a combination of convolution operations, activation functions and

pooling on the data, [34]. In addition to these main processing layers, CNNs also commonly consist of some regulatory operation such as batch normalisation. Furthermore, a CNN can contain some elements that are dependent on the task and data. Fully connected layers and upsampling layers are examples of such elements.

### **Convolution Operations**

The convolution operations involve applying various convolutional filters on the input in order to generate useful spatially correlated features. This is demonstrated by a simple example in Figure 8.

### **Activation Function**

The output from the convolutional operations are passed through a non-linear activation function. This adds non-linearity to the network, which allows the network to compute complex, non-linear semantic features of the input data. Currently, the ReLU activation function and its variations are the most widely used activation functions since they support fast computation and overcome the vanishing gradient problem, [34], [37].

### **Pooling**

The pooling layer is responsible for sub-sampling the data in order to extract the most dominant features of the input. This reduces computation costs and makes the model more invariant to geometrical distortion. There are a range of pooling methods. Max, average and L2 are some examples, [34].

### **Batch Normalisation**

Batch normalisation is also a common feature of modern CNN architectures. Batch normalisation is applied to intermediate layers of the CNN to give input to intermediate layers a fixed distribution. This has a stabilising effect on the training process, making it less sensitive to parameter initialisation and also has a regularising effect on the training process, [38].

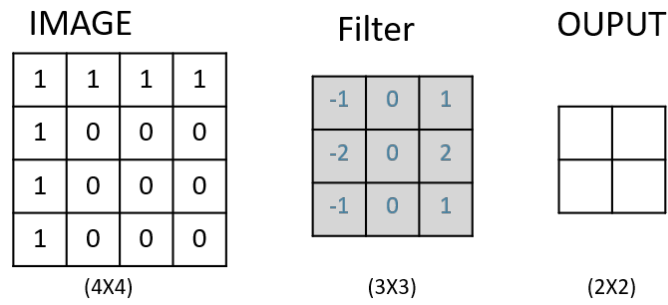
### **Fully Connected Layers**

Fully connected layers are commonly found in the head of a network and their task is to produce the desired output. In fully connected layers all neurons in a layer are connected to all neurons in the preceding and proceeding layers, [39]. For example, in a classification CNN, as shown in Figure 7, the fully connected layer outputs class confidence scores for each of the classes in the dataset.

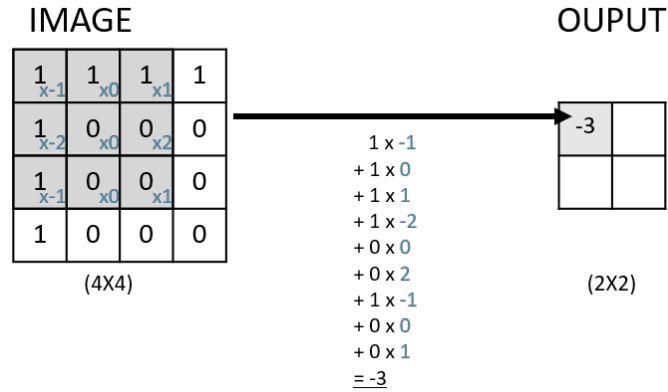
### **Upsampling**

In situations where it is necessary to increase spatial resolution of the data, an upsampling operation is performed. One way of performing upsampling is by a simple 'unpooling' operation, which works by giving the upsampled feature map some interpolated value such as nearest neighbour as demonstrated in Figure 9. However, a more common way to upsample is using an operation called transpose convolution, which is a learnable way of upsampling, [40]. One way of interpreting transpose convolution is that each pixel value of the input image is distributed to its neighbours in the output by multiplying it by a filter. In this way, one input is transformed to many outputs, [41]. The size of the output is determined by the filter size, padding and stride. This is demonstrated by a simple example in Figure 10.

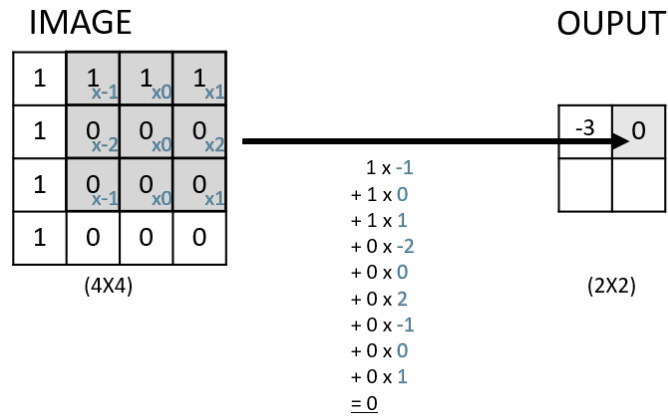




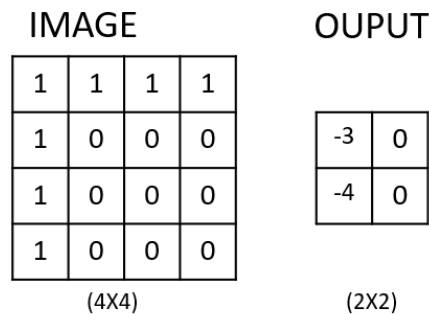
(a) Given a 4x4 image and a 3x3 filter, applying convolution to the image with a stride of 1 will produce an output with the shape 2x2.



(b) Dot multiplication is performed between the filter weights and the image pixel values to produce an output value.

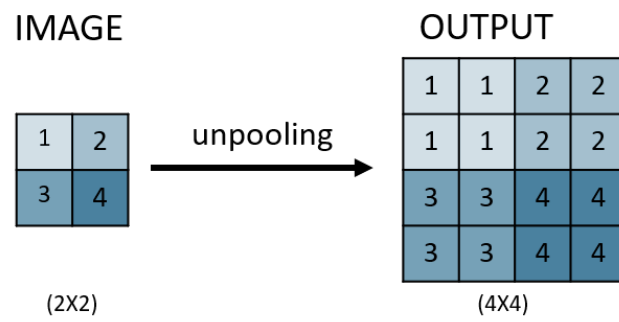


(c) The filter is moved one stride and a new dot product is computed.

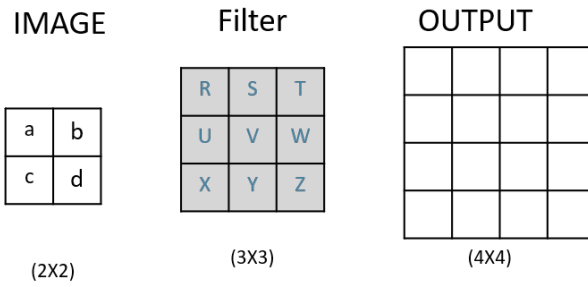


(d) After sliding the filter over the whole image, convolution is complete

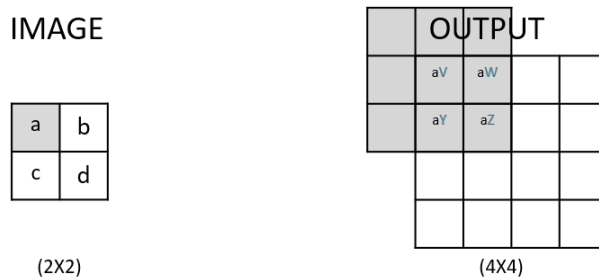
**Figure 8:** Simple example of a convolution operation using a 4x4 image and a 3x3 filter.



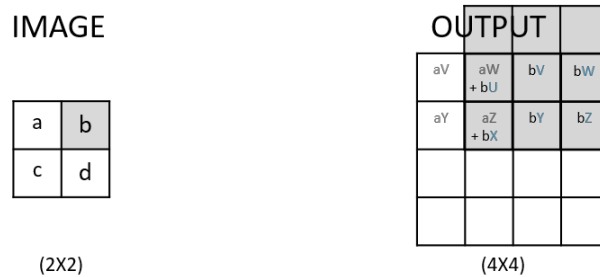
**Figure 9:** Simple example of unpooling by nearest neighbour interpolation



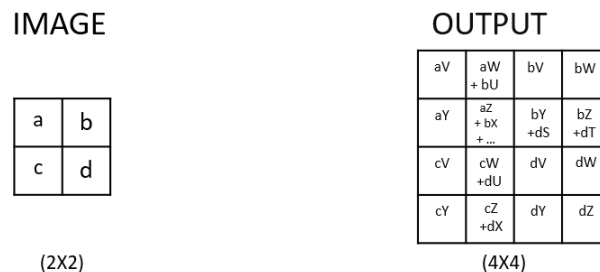
(a) Given a  $2 \times 2$  input image and a  $3 \times 3$  filter, applying transpose convolution with a stride of 2 and output padding of 1 will yield an output with shape  $4 \times 4$



(b) The input pixel is multiplied with the filter to produce values in the output image



(c) The filter is moved two strides and a new dot product is computed. In areas where output values overlap, a sum is computed



(d) After sliding the filter over the whole image, convolution is complete

**Figure 10:** Simple visualisation of a transpose convolution operation using a  $2 \times 2$  image and a  $3 \times 3$  filter with a stride of 2 and output padding of 1.

### 2.4.3 Influential Backbone Architectures

The modern evolution of CNNs for computer vision can be said to have begun in 2012 with Krizhevsky et al.'s AlexNet, which was able to achieve significantly better results on the ImageNet LSVRC-2012 classification contest, [42] than current state of the art (SoTA) methods at the time, [43]. This was possible first in 2012 because graphical processing units (GPUs) had become powerful enough to train deep CNNs on large datasets. Moreover, current datasets such as ImageNet were large enough to avoid overfitting, [43]. Since 2012, access to even greater computing resources as well as larger, more complex datasets have allowed CNNs to become even deeper and wider, which has yielded increasingly more impressive results.

With increasingly deeper network architectures, some issues arise. One issue is the difficulty of custom layer design. As a result, networks have shifted towards more modular, uniform designs, [34]. Another challenge is the vanishing gradient and slow convergence issue. In order to deal with this, various information gating solutions such as ResNet's skip connections and InceptionNet's auxiliary classifiers have been proposed, [34]. In addition, there has recently become a focus on designing light weight architectures that are able to process data in real time whilst still achieving SoTA performance.

This section describes some of the most commonly used recent CNN architectures and their contribution to the field. Visualisations are taken from Raimi Karim's article, 'Illustrated: 10 CNN Architectures' published at Towards Data Science, [4].

#### Inception Net (2014)

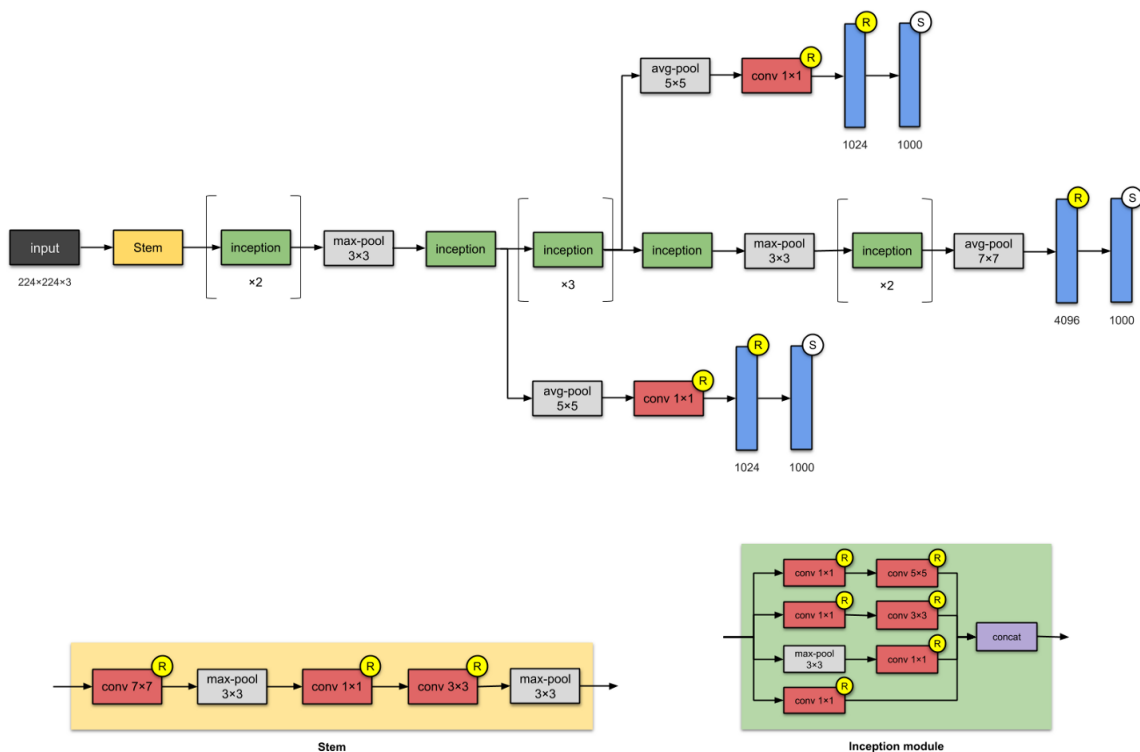
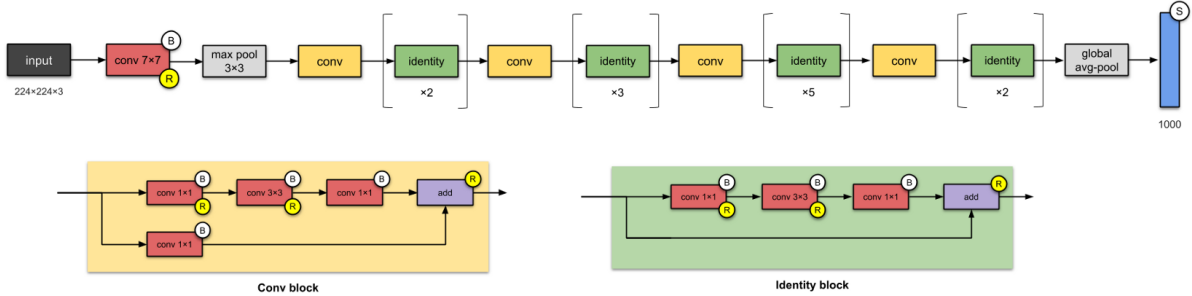


Figure 11: Inception-v1 architecture. Figure origin: [4]

Inception Net, [44] benefits from three main ideas. Firstly, the network is built using so called 'inception modules' as shown in Figure 11. These inception modules are built from parallel convolution branches that are combined using concatenation. In doing this, the network learns to select useful features from each of the parallel branches. The second idea proposed by inception net was the use of  $1 \times 1$  convolutions for dimension reduction. This makes it possible to increase the networks depth and width while not compromising the computational budget. Finally, inception net uses auxiliary classifiers, which has the effect of increasing the

gradient signals during training at earlier stages in the network. These auxiliary classifiers are discarded at inference time.

## ResNet (2015)



**Figure 12:** ResNet-50 architecture. Figure origin: [4]

ResNet, [10] addresses the issue that deep networks are more difficult to train by utilising residual blocks. Residual blocks add skip connections that let layers fit a residual map instead of having to directly fit the underlying mapping. By doing this, information from the previous layer in the network is easily passed to the next layer. Moreover, this greatly reduces the gradient degradation problem.

The architects behind ResNet designed their network with five different depths: 18, 34, 50, 101 and 152. These architectures are shown in Table 2. The deeper variants of the network performed considerably better than its narrower counterparts.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112 × 112	7 × 7, 64, stride 2				
conv2_x	56 × 56	3 × 3 max pool, stride 2				
		$\begin{bmatrix} 3 \times 3, 6 \\ 43 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28 × 28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14 × 14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7 × 7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1 × 1	average pool, 1000-d fc, softmax				
FLOPs		1.8 × 10 <sup>9</sup>	3.6 × 10 <sup>9</sup>	3.8 × 10 <sup>9</sup>	7.6 × 10 <sup>9</sup>	11.3 × 10 <sup>9</sup>

**Table 2:** ResNet architectures. Downsampling is performed by conv3 1, conv4 1, and conv5 1 with a stride of 2. Table origin: [10]

## ResNeXt (2017)

ResNeXt, [45] introduces a new dimension that the authors call cardinality. Cardinality refers to the size of the set of parallel transforms in a block. ResNeXt adopts ResNet’s strategy of repeating layers and residual blocks whilst also exploiting the split-transform-merge strategy of inception net. However, ResNeXt differs from inception net in that parallel paths all have the same topology. ResNeXt is able to improve classification accuracy even when network complexity is maintained.

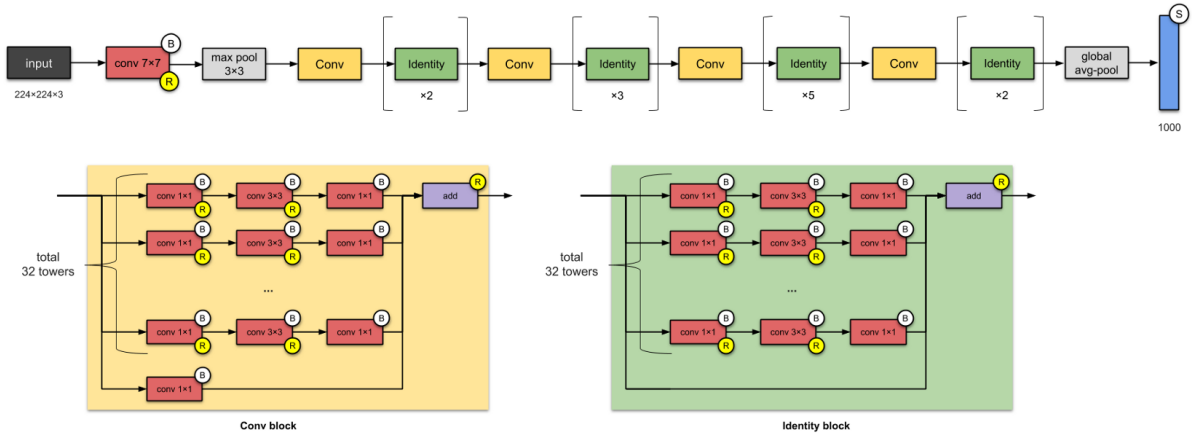


Figure 13: ResNeXt-50 architecture. Figure origin: [4]

## 2.4.4 Training a Neural Network

A CNN is optimised for a specific task by an iterative process called training. This section explains the theory behind the process of training a CNN and the factors that should be considered for this process to be successful.

### Loss Function and Gradient Descent

Training a neural network is done by gradient descent, [46]. Gradient descent is the process of minimising some loss function by iteratively changing parameters in the direction that has the steepest negative gradient in relation to the loss function. In the case of CNNs, the parameters that are adjusted are the weights of the convolutional filter and the bias value. The learning rate defines how much the parameters should be adjusted each gradient descent step.

When gradient descent is done using a subset of the full dataset this is called minibatch stochastic gradient descent. This is commonly how gradient descent is implemented because it is computationally efficient and generalises better than gradient descent based on the full dataset, [47].

The choice of loss function depends on the type of data that is being predicted. If the data being predicted is a continuous value, then a regression loss such as mean squared error should be chosen. On the other hand, for a classification problem, a classification loss such as cross entropy loss is appropriate, [48].

### Overfitting/Underfitting

When training a neural network, the goal is to learn general features of the data that can be applied to make predictions about new unseen data. It is not desirable that the network simply memorise the training data. When the model fits 'too well' to the training data, this is called overfitting, [5]. On the other hand, if the model is too general then this is called underfitting. Figure 14 shows example of over and underfitting.

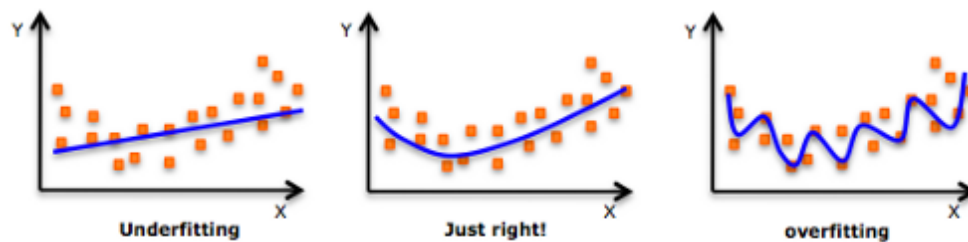


Figure 14: An example of underfitting, overfitting and fitting a model just right. Image origin: [5]

## Dataset Splits - Training, Validation and Test

In order to avoid overfitting, the available data should be split in three parts:

- **Training dataset:** This is the data that will be used to train the network with gradient descent.
- **Validation dataset:** The validation data is used during training to monitor that the network does not overfit on the training data. When training begins, performance of the network on the validation dataset should gradually improve despite the network being fitted to the training set. However, if the network begins to overfit the training data then performance on the validation dataset will begin to stagnate or worsen. At this point it is time to stop training the network.
- **Test dataset:** The test dataset is not used at all during training and as a result it is an unbiased representation of the dataset. After training, the test dataset is used to evaluate the model's performance.

## Transfer Learning

Transfer learning is when network parameters are initialised with weights that have been pretrained on some other large dataset, [49, 50]. Transfer learning is standard practice when working with deep neural networks because it allows the networks to be fine-tuned to the task specific dataset much faster than if the network would be trained from scratch. Moreover, results when using transfer learning are generally better than when starting from scratch, [50]. This is likely due to the fact that models trained on very large datasets will already be very good general feature extractors and the task specific datasets are rarely large enough in themselves to get optimal results, [49].

When using transfer learning to fine-tune a CNN, it can be beneficial to keep some of the pretrained weights of the earlier layers fixed and only update parameters in later layers. This is because earlier layers of CNNs are concerned with more generic features than later layers. Keeping these earlier layers unchanged can also act as a regularising factor and help to avoid overfitting.

## Data Augmentation

In machine learning, access to large amounts of data is crucial for the performance of the model, [51]. If there is not enough data, the model will not generalise well to new unseen data. Data augmentation can be used to increase the variance of the training dataset, which will help to combat overfitting and improve the model performance. Traditional augmentation techniques involve performing a range of random geometric and colour augmentations to the data such as flipping, rotating and random brightness. Generative Adversarial Networks (GANs) provide a new method of increasing the size of a dataset as they are able to generate new images from the training dataset, [51].

## Deciding Training Parameters

Success of deep learning is dependent on choosing good hyperparameters. Two of the most crucial hyperparameters are the choice of learning rate and batch size.

The learning rate controls how much the parameters should be adjusted in each iteration of gradient descent. As a result, choosing an appropriate learning rate is crucial for the success of the training process. If the learning rate is too low, training progression will be slow and there is a risk of getting stuck at a local minimum. On the other hand, if the learning rate is too high, the training process will be very turbulent and there is a risk of overshooting the target, [52]. Unfortunately, learning rate cannot be known in advanced and must be found through a process of trial and error, [52]. One way to do this is to do a grid search of a range of learning rates and for each learning rate observe the training loss over time. An appropriate learning rate will have a loss curve that converges in reasonable time, begins to plateau at a loss value that is as low as possible and does not oscillate too much, [52].

The use of momentum and learning rate scheduling are methods of altering the size of the gradient descent step during training. Learning rate scheduling involves varying the learning rate over the training process. Typically, this involves gradually decreasing the learning rate over the training process as the network gets closer the solution and is gradually making smaller fine-tuning changes.

Momentum is when past updates contribute to the direction of the next update by an exponentially decaying moving average of past gradients. This has the effect of accelerating learning and can help to push the learning process out of local minimum situations, [52]. There exists some adaptive optimisation strategies that apply momentum and adaptive learning rate scheduling. Adaptive Moment Estimation (Adam) is an example of a popular optimisation algorithm that computes individual adaptive learning rates for different parameters, [53]. This makes training progression faster. However, despite its popularity, recent papers have noted that Adam generalises worse compared to simple stochastic gradient descent with momentum.

Another important hyper parameter is batch size. If multiple GPUs with sufficient memory are available, increasing batch size by parallel processing is an effective way of reducing training time, [54]. On the other hand, the use of large batch sizes have been observed to reduce the generalisation ability of the network because a greater proportion of the training data is used for each backpropagation step. When changing the batch size, research suggests a linear scaling strategy for the learning rate so that each image has the same influence to the parameter update process, [55, 54]. In practice this means that when batch size is multiplied by  $K$ , multiply the learning rate by  $K$ . When working with linear scaling of learning rate with regards to batch size, a warm up strategy can be beneficial because this will apply lower learning rates in the beginning of training when the network is changing rapidly, [55].

### 2.4.5 Influential Architectures for Object Detection

The field of computer vision and object detection is currently evolving at a fast pace and new innovative solutions are continuing to top the charts of object detection challenges such as the COCO detection challenge, [56]. Object detection using CNNs can be grouped into two categories: multi-stage detectors and single stage-detectors, [30]. The R-CNN family of algorithm is an example of a multi-stage detector and it involves two steps: Region proposal and classification. In contrast, Single-stage detectors skip the region proposal stage and run detections in just one stage. YOLO, SSD and RetinaNet are examples of single stage detectors. In general, multi-stage detectors often achieve slightly better detection performance, while single-stage detectors are more time efficient, [57]. This section presents some of the more influential deep learning-based architectures for object detection.

#### R-CNN (2014)

R-CNN, [58] was among the first algorithms to use deep learning for object detection and the idea behind it was simple. First, 2000 regions are proposed using a classical region proposal technique called selective search. Each of these regions are then warped to a fixed size and fed to a CNN. Finally, the output from the CNN is fed to a linear Support-vector machine classifier, which determines the object class.

At the time of release, R-CNN improved previous best mAP results on the VOC2012 dataset, [59] by more than 30%. However, the algorithm had some clear disadvantages. The main disadvantage was that it was very slow because the 2000 regions were processed independently through the CNN despite having a great deal of overlapping features.

#### Fast R-CNN (2015)

In 2015, R. Girshik proposed Fast R-CNN, [60], which addressed the main issues of R-CNN by processing each image only once through the CNN and then projecting the regions proposed by selective search onto the already computed feature map. Finally, each region of the feature map was sent to two fully connected layers that output the class scores and bounding box regressions.

Fast R-CNN was 10x faster to train than the original R-CNN. At test time, it could process an image in 2.3 seconds. In Fast R-CNN, the processing time was bottlenecked by the selective search region proposal stage.



### Faster R-CNN (2015)

In 2015 S. Ren *et al.* proposed Faster R-CNN, [61], which addressed the main issue of Fast R-CNN by introducing a region proposal network (RPN) to replace the expensive selective search process.

This was very effective since the RPN was also able to profit from the convolutional features generated by CNN backbone. Faster R-CNN reduced proposal time from 2.3 seconds to only 0.2 seconds and also improved mAP performance.

### You Only Look Once (YOLO v1 2015, v2 2016, v3 2018)

YOLO, [62] was proposed by J. Redmon *et al.* in 2015 and was the first single-stage deep learning-based detector. In YOLO, bounding boxes and class probabilities are predicted directly by a single CNN. YOLO works by splitting the image into an  $S \times S$  grid and for each grid cell, B bounding boxes and corresponding class probabilities are predicted. The greatest advantage of YOLO is that it is very fast, and it generalises well. Another general advantage of single-stage detectors over multistage detectors is that they 'see' the entire image, which allows them to reason globally and encode contextual information about the image. Because of this, YOLO demonstrates fewer false positives. The original YOLO had some issues. Each grid cell could only predict one object and as a result, the algorithm struggled to detect small objects that were very close to each other. Compared to existing region-based methods, YOLO had a higher localisation error and a lower Recall.

In 2016, J. Redmon proposed YOLOv2, [63], which introduced some improvements to the original YOLO. The main improvements of YOLOv2 was adding batch Normalization layers, a slightly deeper network and the use of Anchor Boxes. YOLOv2 was better and faster than the original but still struggled with small objects. YOLOv3, [16], released in 2018 by the same authors dealt with the problem of detecting small objects by implementing multi-scale detection with a process similar to Feature Pyramid Network. In addition, YOLOv3 used a deeper network than YOLOv2. Despite a deeper network and multi-scale predictions, YOLOv3 is still very fast. YOLOv3 is about 3.8x faster than its counterpart RetinaNet but still achieves very similar  $mAP_{50}$  score. YOLOv3 certainly has a great speed-accuracy trade-off and is therefore a very popular method for applications that require real time processing. On the other hand, YOLOv3 struggles to get boxes perfectly aligned, so it does achieve a lower  $mAP_{0.5:0.95}$  score than other SoTA detectors.

### Feature Pyramid Networks (2016)

In 2016, T.-Y. Lin *et al.* proposed using Feature Pyramid Networks (FPN) for Object Detection, [64]. FPN exploits the natural pyramid structure of CNNs in order to improve detection over multiple scales. In general, this is done by combining low-resolution, semantically weak features with high-resolution, semantically strong features in order to generate semantically strong features with high resolution. By using FPN with a basic Faster R-CNN system, the authors were able to achieve better performance on the COCO detection challenge than all other existing single-model entries. Most of the SoTA systems today (both multi-stage and single-stage) incorporate FPN.

### Cascade R-CNN (2017)

In 2017, Z. Cai *et al.* proposed cascade R-CNN, [17]. Cascade R-CNN is a multi-stage detector that involves training a sequence of detectors with increasing IoU thresholds. This has the effect of improving the quality of detection for all IoU values but especially for stricter IoU levels. The architecture achieved SoTA on the COCO detection challenge. The authors propose that this cascading architecture is applicable and advantageous across detector architectures and not just for R-CNN.

### RetinaNet (2017)

In 2017, T.-Y. Lin *et al.* proposed RetinaNet, [18]. The designers of RetinaNet identified that extreme foreground-background imbalance during training was a cause for why single stage detectors had previously achieved inferior accuracy compared to multi-stage detectors. RetinaNet deals with this problem by

introducing Focal Loss, which weights down the loss for well classified examples. In addition, RetinaNet also utilises FPN in its design. In doing this, RetinaNet was able to match the speed of other single-stage detectors and surpass the accuracy of all other existing SoTA detectors.

### Fully Convolutional One-Stage Object Detection (FCOS) (2019)

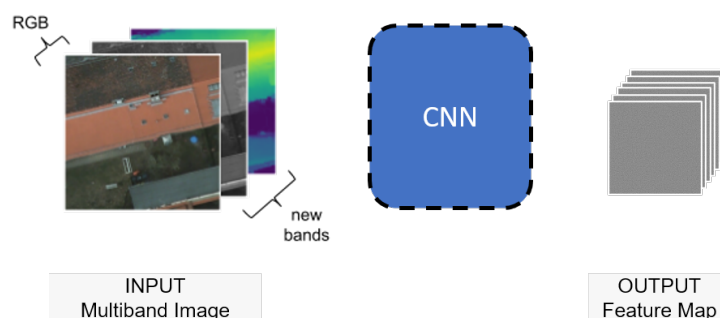
In 2019, Z. Tian *et al.* proposed FCOS [65]. FCOS is different from other SoTA detectors since it is anchor box free. Instead of anchor boxes, FCOS outputs a 4D vector for each pixel in the image, which describes the pixels position relative to the edge of a bounding box. The advantage of this is that complicated calculations such as IoU are not needed. Moreover, tuning of hyper parameters related to anchor-boxes are also not needed. The authors report that FCOS surpasses performance of previous single stage detectors with the advantage of being simpler.

### Libra R-CNN (2019)

Libra R-CNN, [19] was proposed in 2019 by J. Pang *et al.* Libra R-CNN is a framework that aims to balance the training process in three stages: sampling stage, feature stage and objective stage. By doing this, detection mAP performance is improved significantly compared to FPN Faster R-CNN. The framework was also successfully applied on the single stage architecture RetinaNet but then without the balanced sampling procedure (as retinaNet does not have a sampling step).

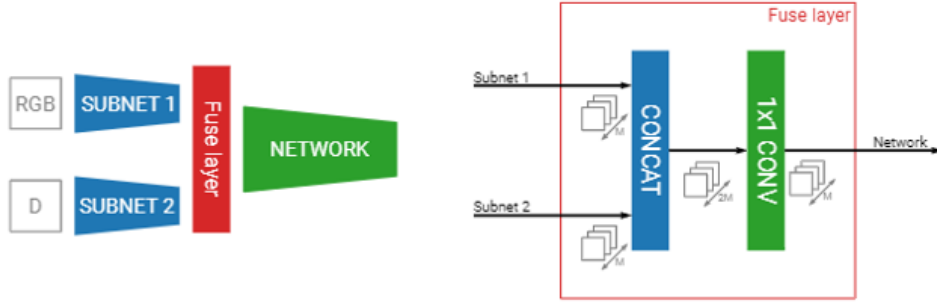
## 2.5 Multi-channel Image Fusion

There are many datasets that have multiple channels in addition to the regular 3-channel RGB image. For example, the ISPRS potsdam dataset, [66] contains satellite images with 5 channels (red, green, blue, infrared and elevation). Another such dataset is RGB + Depth images used for pedestrian detection, [67]. It is clear that these extra channels contain valuable information for detection and segmentation purposes, however the challenge is to find the best way to include this extra information in a deep learning model. Given that  $N$  channels are available, one way to include these extra channels is to simply change the CNN model to accept  $N$  input channels instead of 3 as shown in Figure 15. The problem with this solution is that having an input with anything other than 3 channels means transfer learning is no longer possible because the pretrained models used for transfer learning are based on 3 channel input, [6]. As a result, 3-channel RGB only networks often outperform models trained from scratch with multiple input channels, [6].



**Figure 15:** Extending CNN model to accept multiple input channels. Image adapted from [6]

One solution that has successfully been used to utilise multi-channel input whilst maintaining the advantages of transfer learning is to run the various inputs through parallel subnetworks and later fuse the feature maps of the two subnetworks somewhere in the middle of the network. This architecture is demonstrated in Figure 16. Fusion is performed by concatenation followed by  $1 \times 1$  convolution to restore the original depth of the network. In this way, the  $1 \times 1$  convolutions should ideally learn to extract features from both subnetworks to create a richer combined feature map.



**Figure 16:** Fusion Network - a method of taking advantage of multiple input channels whilst keeping the advantages of transfer learning. The two input images are first run through two parallel subnetworks who's outputs are later fused somewhere midways in the network. Image origin: [7]

In [6], Hassan experimented on using this kind of fusion network for segmentation on RGB + Elevation satellite images. A ResNet backbone was used and results from fusion networks with fusion at various depths in the ResNet backbone were compared against results from an RGB only network and 4-channel input RGBE network trained from scratch. Results showed that the fusion networks outperformed both the RGB only network and the RGBE network for every class. Fusion at mid to late layers in the ResNet backbone gave the best results with slight variation from class to class.

A similar experiment was conducted by Ophoff, Beek and Goedeme in [7], where they conducted an extensive experiment on the best place to perform fusion in YOLO v2's DarkNet for the task of object detection with RGB + Depth images. The fusion networks again outperformed the RGB only networks and it was found the mid to late fusion gave the best results, again with some variation by class.

## 2.6 Camera Calibration

The goal of camera calibration is to calculate intrinsic and extrinsic camera parameters in order to undistort and align images with real-world coordinate systems, [68]. Intrinsic camera parameters are parameters related to the camera such as focal length and lens distortion. On the other hand, extrinsic parameters are those parameters that describe the cameras pose in relation to the real-world, [69]. Camera calibration is helpful in correcting distortion effects in images as well as finding the mathematical relationship between two different coordinate system. Using this, alignment of two images of the same scene can be performed.

### 2.6.1 Lens Distortion

There are two main types of lens distortion that typically occur in images. These are radial and tangential distortion. Radial distortion is the fish-eye like effect that causes straight lines to appear curved. Tangential distortion occurs in an image if the lens is not perfectly parallel to the imaging plane and creates the effect of making some areas of the image appear larger than expected, [70].

Brown's distortion model, [71] gives a formula for correcting an image for both radial and tangential distortion, [72]. Brown's model is expressed by equations

$$x' = x + (x - x_c)(k_1 r^2 + k_2 r^4 + \dots + K_n r^{2n}) + (P_1(r^2 + 2(x - x_c)^2) + 2P_2(x - x_c)(y - y_c))(1 + P_3 r^2 + \dots + P_m r^{2m}) \quad (2.3)$$

and

$$y' = y + (y - y_c)(k_1 r^2 + k_2 r^4 + \dots + K_n r^{2n}) + (2P_1(x - x_c)(y - y_c) + P_2(r^2 + 2(y - y_c)^2))(1 + P_3 r^2 + \dots + P_m r^{2m}), \quad (2.4)$$

where  $(x', y')$  are the undistorted image coordinates,  $(x, y)$  are the distorted image coordinates,  $(x_c, y_c)$  is the images principle point,  $r$  is the distance of a pixel from the image's principle point,  $K_n$  is the  $n^{th}$  calibration coefficients and  $P_m$  is the  $m^{th}$  tangential distortion coefficient.

Brown's distortion model has 9 unknown variables, describing internal camera parameters. These are the camera's principle point  $((x_c, y_c))$  and focal lengths  $((f_x, f_y))$  [70], which are expressed in the camera matrix (C):

$$C = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \quad (2.5)$$

and the 5 distortion coefficients:

$$\text{Distortion coefficients} = (K_1, K_2, K_3, P_1, P_2). \quad (2.6)$$

In order to solve for the camera matrix (C), the pinhole model can be used. The pinhole model describes the relationship between the homogeneous coordinates of a 2D point in the image coordinate system  $([x, y, 1]^T)$  and the corresponding homogeneous coordinates of a 3D point in the real-world coordinate system  $([X, Y, Z, 1]^T)$  assuming no distortion, [73]. This relationship can be used to solve for the camera matrix (C) and extrinsic parameters  $((R, t, s))$ . The pinhole model is described in equation 2.7.

$$s \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = C[Rt] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \quad (2.7)$$

Given the pinhole model and minimum 5 known corresponding 2D image and 3D world points, a homogeneous linear system can be set up and used to solve for the camera matrix (C) and the extrinsic parameters  $(R, t, s)$ , [74].

Zhang, [73] suggests to simplify this model further by observing a planar pattern rather than a real-world 3D object. In this case, the pinhole model will map a point in 2D space to another point in 2D space, which reduces the number of unknowns to be solved for. Moreover, this method greatly simplifies the calibration setup because precise knowledge of 3D world coordinates is not necessary.

In dealing with distortion, Zhang suggests an iterative approach, [73]. Assuming relatively small distortions, a good initial estimation for the camera matrix can be found using the pinhole model without accounting for distortion. The values of the camera matrix can then be used in Brown's distortion model (equations 2.3 and 2.4) to estimate the distortion coefficients, which can be used to estimate new undistorted coefficients. This process is repeated but using the new undistorted coordinates rather than the original image points. Several iterations of this process should eventually lead to convergence of parameters.

## 2.6.2 Affine Transformation

The extrinsic parameters describe the rotation, scaling and translation transformations that map the coordinates of an image to its corresponding real-world coordinates. Rotation, scaling and translation are affine transformations since they preserve straight lines, [75]. If real-world coordinates are simplified to a 2D space and the transformations are written in homogeneous form, the 3x3 affine transformation matrix (T) that maps a 2D homogeneous image point  $([x, y, 1]^T)$  to a 2D homogeneous real-world point  $([X, Y, 1]^T)$  can be expressed as in equation

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = T \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}, \quad (2.8)$$

, [75]. Given a set of corresponding points in the two coordinate systems, the least square method can be used on the relationship described in equation 2.8 to estimate  $T$ .

## 2.7 Summary

The theory and background information presented in this chapter provides the foundation used to motivate, develop and evaluate a deep learning-based model that automatically detects and locates sheep in multi-channel UAV images.

Sheep grazing and roundup is the process that this thesis aims to improve. Keeping an overview of free grazing sheep and effective rounding up of sheep at the end of the grazing season can be a challenge due to the vast and rugged terrain that the sheep disperse themselves across throughout the grazing season. Luckily, there exists some technologies on the market today that help improve sheep oversight. These are: bells, radio-bells, electronic ear tags and UAVs. Bells are limited since they are only useful for finding sheep when the farmer is already nearby. Similarly, electronic ear tags require the sheep to be in close contact with a sensor for registration to occur. Radio-bells are a more modern technology that aids in controlling and locating sheep by GPS tracking. However, radio-bells are also limited by their high unit price and (in most cases) mobile coverage requirement. Finally, UAVs can be a useful tool for finding sheep since they allow the user to effortlessly survey areas of challenging terrain without physically having to walk there. Moreover, commercial camera UAVs are becoming increasingly more affordable. However, sheep detection in UAV footage is currently a manual process, which can be time consuming, boring and error prone especially if this is done live in the field on a small mobile screen. As a result, automated sheep detection in UAV imagery as proposed in this thesis can be a useful addition to the current technology available to sheep farmers.

Previous master theses and projects, [2, 21, 22, 23] have explored the task of automatic sheep detection in UAV images using both traditional and deep learning-based approaches. All these attempts have looked at either RGB or infrared images alone and come to the conclusion that there is a high potential performance gain to be had from combining information from both image types. In addition, previous research has been limited by small datasets and conclude that a larger more realistic dataset is required especially for a deep learning-based solution. This thesis builds on previous attempts by using both RGB and infrared imagery and by having a larger dataset including some free ranging sheep.

The task of detecting and locating objects in an image is called object detection. Most commonly, object detection solutions aim to predict bounding boxes for each instance of certain objects in an image. The quality of an object detector can be judged by its precision and recall score on a test dataset. Precision is the proportion of detections made that are correct detections whilst recall is the proportion of all ground truth instances detected by the object detector. Average precision takes both precision and recall into account and is therefore the very common metric used to evaluate the performance of an object detector.

Currently, deep learning-based approaches using convolutional neural networks (CNNs) are topping the charts of the largest object detection and other computer vision task competitions in the world. New innovations and tweaks are continuing to improve the performance of these CNNs but the underlying building blocks remain constant. A CNN used for computer vision is made up of a backbone, a neck and a head. The backbone is responsible for computing features of the image, the head is responsible for converting the feature maps to the desired output (e.g. bounding boxes ) and finally the neck is responsible for connecting the network backbone and head with each another. These building blocks themselves are built from many layers that are responsible for operations including convolutions, activations, pooling and upsampling. Influential backbone architectures such as ResNet or ResNeXt are constructed by organising these layers and operations in a specific way that maximises the feature generation capability. Sections 2.4.4 and 2.4.5 elaborate further on the considerations that go into successfully training a CNN as well as influential CNN architectures for the specific task of object detection, which provides the basis for making the neural network architectural design choices for the specific task of sheep detection in multi-channel images.

Previous research on designing CNNs to accept multi-channel input beyond regular 3-channel RGB imagery, [6, 7] report that the naive approach of simply stacking the new channels on top of the RGB channels and altering the first CNN layer to accept extra channels is not ideal. This is because pretrained

networks commonly used to warm start training of neural networks are all pretrained on 3-channel RGB images. Instead, successful integration of additional input channels has been achieved by first processing the inputs separately then performing fusion of the feature maps somewhere in the middle to late layers of the CNN architecture. This way, the network is able to benefit from the valuable information present in the extra channels of data while still maintaining the benefits of transfer learning.

Another challenge that may be encountered with multi-channel input is misalignment between image pairs due to distortion and varying image resolutions. Image alignment can be performed in a two-step process:

1. correct images for distortion
2. perform an affine transformation on one of the images to scale, translate and rotate it relative to the target image.

Correcting for lens distortion entails finding a set of distortion coefficients and internal camera coefficients. Zhang's, [73] approach to finding these coefficients involves observing a set of photographs of a planar pattern such as a checker-board and iteratively alternating between using the observed corner points to estimate the camera coefficients using the pinhole model (Equation 2.7) and the distortion coefficients (Equation 2.6) using Brown's model, [71]. After correcting for distortion, the affine transformation between the two image coordinate systems can be found using the least squares method on a set of corresponding points on the two distortion free images.

### 3 Project Description

This chapter presents an overview of the proposed system that aims to improve sheep round up using multi-channel UAV mapping and deep learning. Furthermore, the requirements to the image processing part of this system are explained, and the properties of the available data and hardware used for the thesis is presented.

#### 3.1 Envisioned Solution

A system that uses a UAV and automatic sheep detection in images is suggested as a solution to aid farmers in locating sheep at the end of the grazing season. This system involves two modules:

##### Real-Time Module

A mini-computer mounted on the drone will process the data as it is captured. The user can see results in real-time through a user interface. Since the real time module is limited by time constraints and available processing power, result accuracy is also limited. This module is useful for getting results fast when the user is out in the field.

##### Post-Processing Module

In the post processing module, data will be processed on external servers, which will allow for more heavy-weight models and thus higher quality results. This module is useful when analysing large areas in advance of going into the field to manually search for the sheep as it can give a good indication of where to begin the search.

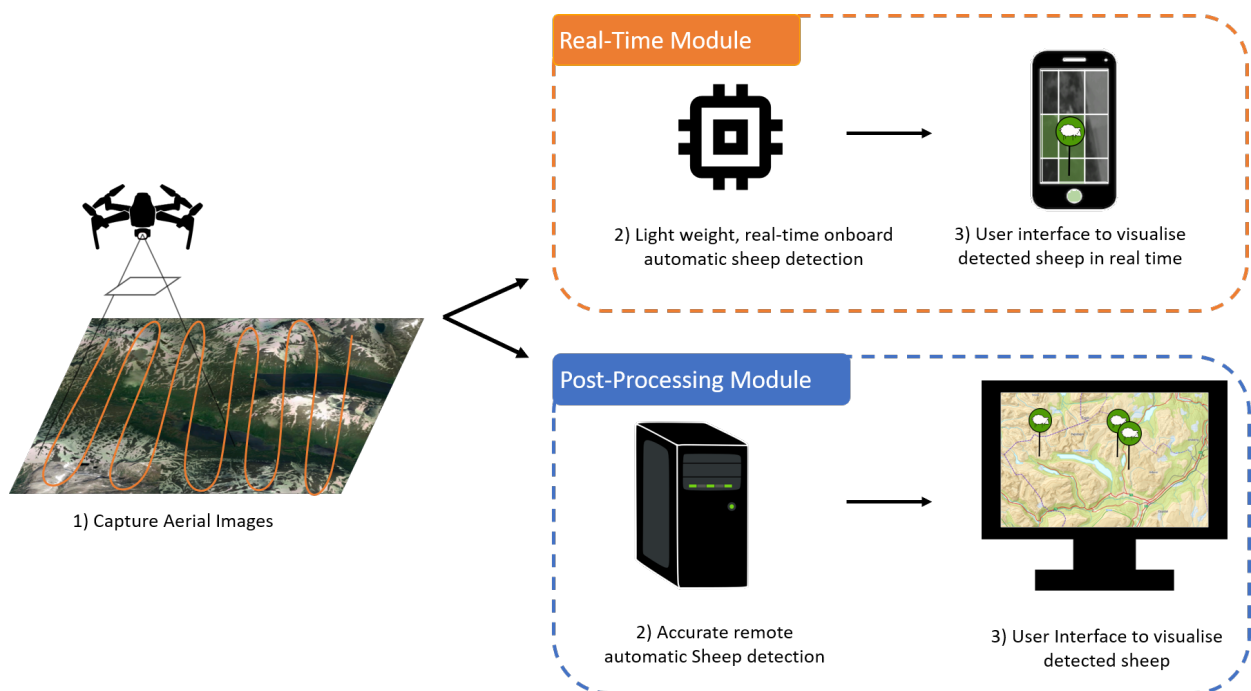
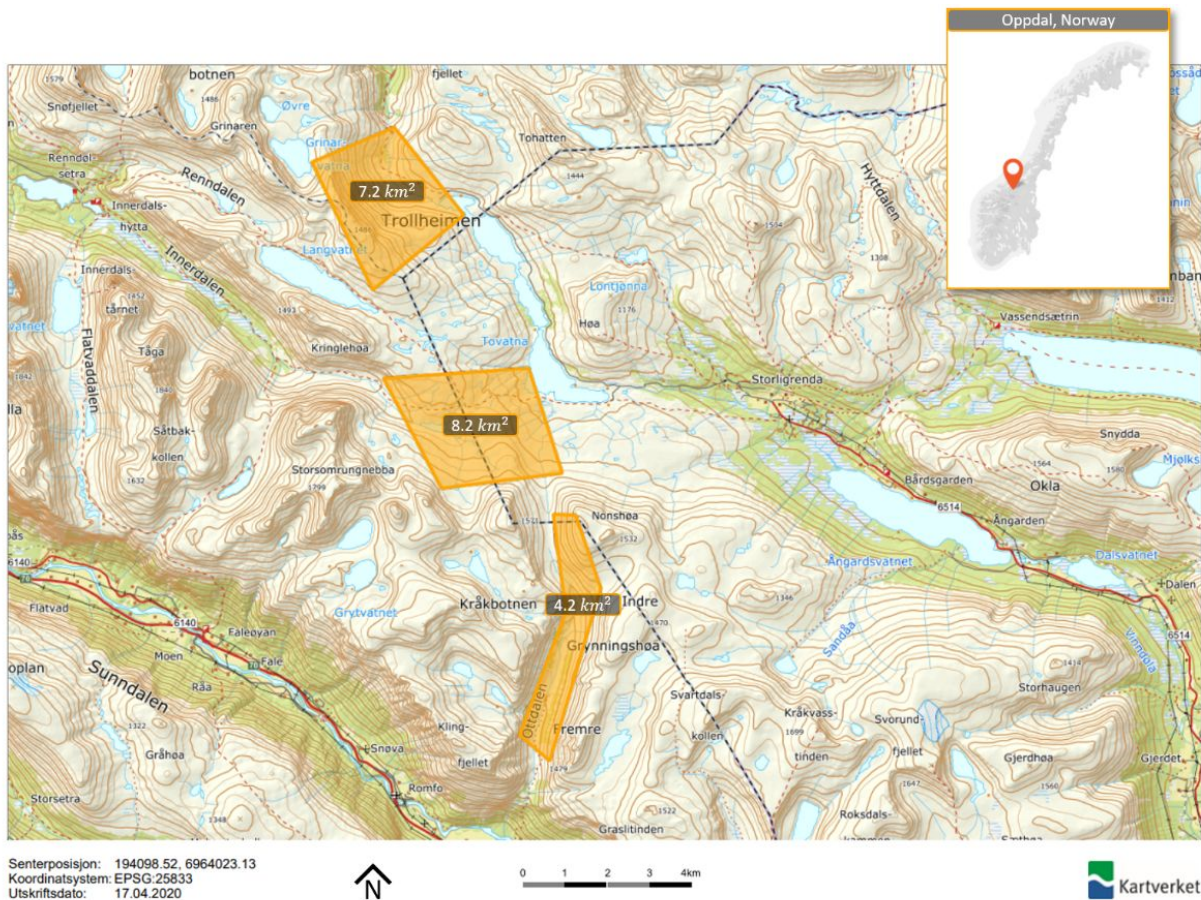


Figure 17: Envisioned system to find sheep.

Figure 18 shows some sample of search areas where the post processing module of the aforementioned solution could be applicable. The marked areas are some examples of areas where Hvasshovd has previously

experienced finding straggling sheep after the initial round of sheep collection. These areas are also characterised by long distances and challenging terrain, which makes manual search with binoculars challenging. Moreover, these areas are approximately twice as large as an area one could expect to perform a manual search in one day. In later phases of the sheep round up, it may also be relevant to search even larger areas.



**Figure 18:** Sample search areas around Storlidalen in Oppdal

This system has some advantages over current technologies since it does not rely on mobile coverage and the sheep do not have to wear an expensive tracker. On the other hand, this system does have some vulnerabilities. One issue is that detection likelihood is reduced if sheep are occluded by other objects such as trees. Moreover, image quality is dependent on weather conditions. For example, precipitation and strong winds create unsuitable flying conditions for most UAVs and too much sun will reduce the image quality. In addition, object detection algorithms are quite good but they are not perfect so there is a risk of finding false positives (for instance mistakenly classifying a rock as a sheep). Finally, although small, low end dual camera UAVs are relatively affordable, these are most likely not suitable for mapping such large areas as those exemplified in Figure 18. As a result, the cost of such an operation could be an issue.

The strength of this system compensates for some weaknesses of existing solutions, however it does have weaknesses of its own. As a result, it is imagined that this solution could be used in combination with existing technologies.

## 3.2 Requirements

In this project, detection using deep learning is applied to UAV images of sheep. This detection is envisioned to be used to detect sheep from aerial imagery of potential areas where sheep are grazing in order to help farmers locate their sheep for collection at the end of the season. Further description of the envisioned system can be found in section 3.1 of this report. This thesis focuses on the image processing aspect of



the solution and not data collection or the creation of a user application. With this use case in mind, some requirements for the sheep detector can be set. An overview of these requirements can be seen in Table 3.

### **R1: Localisation Quality**

Localisation quality refers to how accurate the localisation of the sheep should be. A perfectly fitting bounding box or segmentation around each sheep would be a very high localisation quality. This is not required for the desired use case. For bounding boxes, a course fitting box with intersection over union (IoU) greater than at least 0.5 is more than enough. In fact, for the desired use case it is sufficient for the detector to predict the presence of sheep within a fixed area of 50x50m.

### **R2: Precision and Recall**

The detector should aim to maximise precision and recall. Precision and recall are explained in more detail in section 2.3.1. In short, precision describes how many of the predicted sheep are actually sheep and recall describes what proportion of the total number of sheep in the data set were found by the detector. Average precision (AP) is the average precision over a range of recall values and is therefore a suitable metric that takes both precision and recall into account.

### **R3: Time**

Time is an important requirement to consider and should be seen in perspective with the desired use case, average precision requirements and technological constraints. Since measures to reduce processing time often have some extent of negative impact on precision, recall and localisation quality, an acceptable trade-off must be made. In the final solution, it should be possible for the users themselves to decide between either a fast but slightly less accurate model or a more accurate but slower model. For the solution envisioned, there is a real-time module and a post-processing module.

For the real-time module, images should be processed rapidly and in real-time. 0.11 seconds per image or less would be ideal because this will keep pace with the video frame rate of the thermal camera used, [76]. However, a processing time of up to 0.5 seconds per image would also be fine and give a relatively smooth user experience.

For the Post-Processing module, real-time processing is not required, and processing can be distributed to multiple external servers. Nonetheless processing time should be 'reasonable' and as low as possible since it is costly to rent multiple suitable machines.

In order to get some indication of what is considered a reasonable processing time, a sample use case is considered. This use-case considers the sample search area of  $8.2km^2$  shown in Figure 18. Hvasshovd suggests that a processing time of approximately 30 minutes would be acceptable for processing images covering this area.

Mapping of this sample area with the same camera and ground sampling distance as the available data used in this project, would require approximately 9500 images according to the DJI Pilot app mission planning software. This is with a front and side overlap of 20% between images to ensure that the whole area is mapped. If the constraint of 30 minutes is set to the post processing time then this would require a processing time of approximately 0.2 seconds per image, which would in fact be an acceptable real-time processing speed. However, longer processing times are acceptable since processing can be distributed. A linear increase in processing time can be expected for each additional machine.

Finally, it should be noted that time for data-collection and data-transfer are also important factors for a complete system, however only processing time of the detector is considered in this thesis.

### **R4: Image Quality**

Since image quality can vary based on factors such weather conditions and UAV movement, the object detector should be robust against variances in image sharpness, brightness and contrast.

## R5: Sheep variation

In Norway, sheep come in a range of shapes, sizes and colour. Figure 19 shows some examples of how the different races of sheep can appear. As a result, the object detector should be robust to variation in sheep shape, size, race and colour.



**Figure 19:** Some examples of sheep races that exist in Norway, [8].

## R6: Terrain/background Variation

Sheep can be found in various terrain. The object detector should be able to find the sheep regardless of where they are. This includes rocky areas, area with many trees, on the road, in the snow or anywhere else where the sheep may be seen from the UAV.

## R7: Other animals

The object detector should be able to distinguish sheep from other animals that could be found in the area of interest. Examples of other animals that can be expected to be found in the same areas as the sheep are: cows, dogs, fox, moose and reindeer. Humans can also be found in these areas.

ID	Requirement
R1	The detector should be able to detect the presence of sheep within a fixed area of 5x5m
R2	The detector should aim to maximise Average Precision
R3	The detector should process an image within a reasonable time
R4	The detector should be robust with regards to sharpness brightness and contrast
R5	The detector should be robust to variation in sheep shape, size and colour
R6	The detector should be able to detect sheep regardless of terrain, background or weather.
R7	The detector should be able to distinguish sheep from other animals in the areas of interest

**Table 3:** Object Detection Requirements.

### 3.3 Sheep Dataset

The data used for training and evaluating the sheep detection model is a set of UAV images of sheep captured in 25 separate sessions in August, September and October of 2019 as well as May of 2020. The images captured in 2019 are all taken in Storlidalen valley in Oppdal and feature a mix of Norwegian White Sheep and Norwegian Pelssau. The images from 2019 are the same images used in the project preceding this thesis, [23]. The images from May 2020 are captured in the areas Klæbu and Orkanger and feature some Norwegian White Sheep and some Old Norwegian Spælsau. Images from May 2020 also differ from the 2019 images by featuring an abundance of lamb.

Infrared and RGB image pairs are captured using the dual camera UAV DJI Mavic2 Enterprise Dual (M2ED) and labelled with bounding boxes using the labelling tool Lablebox [77]. This chapter gives an overview of the labelled dataset and its qualities. A sample of the images can be seen in Figure 23.

The dataset consists of 1005 image pairs, which contain a total of 8413 labelled sheep. The sheep in the images vary in size mainly due to images having been captured at different flight heights. In this thesis, images are grouped by the median adult sheep length ( $D$ ) as a proxy measure for flight height. These size classifications are shown in Figure 20. Using the fact that the average adult sheep is 1.3m long, [9], a ground sampling distance (GSD) in cm/pixel can be estimated from the depicted sheep size in pixels (equation 3.1). Due to the linear relationship between flight height ( $H$ ) and GSD, this can also be used to estimate  $H$ . The relationship between GSD and  $H$ , shown in equation 3.2 is obtained using simple linear regression on values given by the DJI pilot mission planning software for the particular UAV used.

$$GSD \approx \frac{130}{D} \quad (3.1)$$

$$H \approx 32.725 \times GSD \quad (3.2)$$

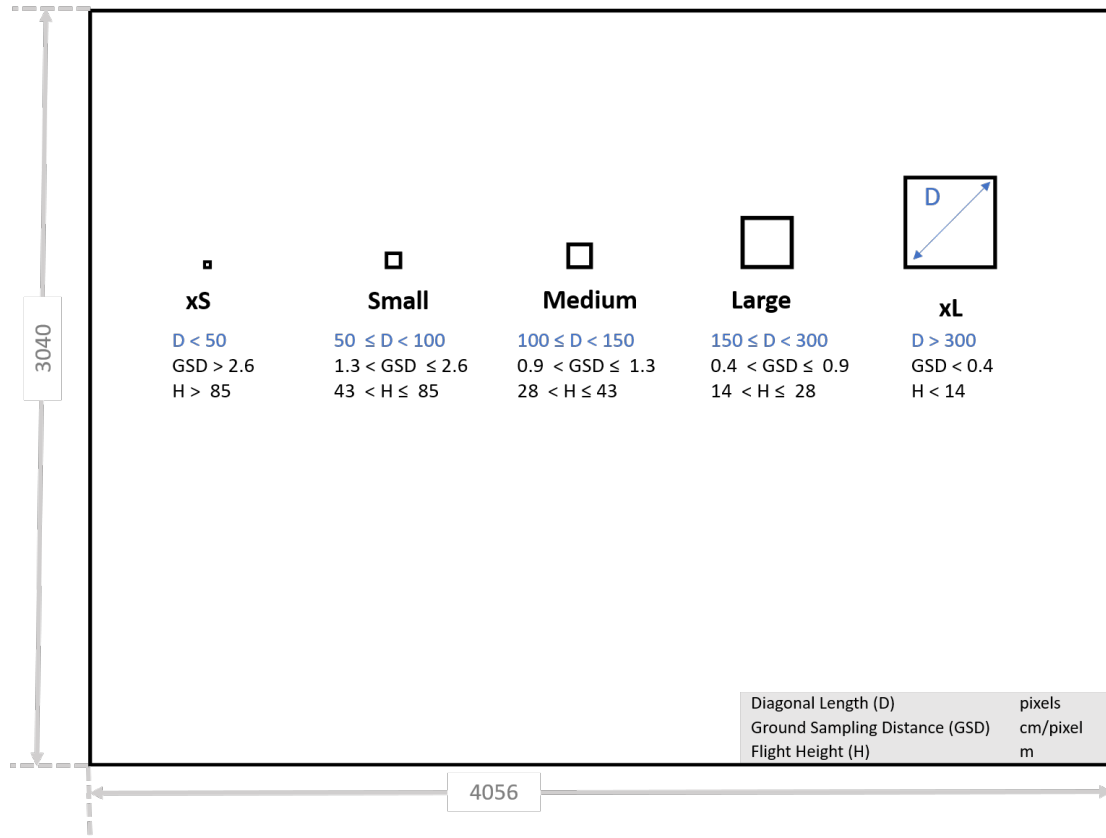
Other differences found in the dataset are grazing location and infrared image type. Some of the images are of free ranging sheep whilst others show sheep that are grazing in an open, fenced area. Moreover, some of the infrared images are in Multi-Spectral Dynamic Imaging (MSX) format whereas others are regular infrared images. An overview of the number of images grouped by month, distribution of sheep size, distribution of free ranging versus fenced sheep and distribution of MSX or regular infrared is shown in Table 4.

	Total	Environment		Median Adult Sheep Size					Infrared Type		Sheep Races*			Background only
		Free	Fenced	xS	S	M	L	xL	Normal	MSX	1	2	3	
August	592	592	0	109	297	91	91	4	434	158	✓	✓		0
September	116	67	49	0	29	52	31	4	7	109	✓	✓		0
October	75	0	75	0	2	27	39	7	1	74	✓	✓		0
May	222	0	200	0	75	83	42	0	222	0	✓		✓	22
All Labelled	1005	659	346	109	403	253	203	15	664	341	✓	✓	✓	22

**Table 4:** The number of images in the sheep dataset grouped by month, distribution of sheep size, distribution of free ranging versus fenced sheep and distribution of MSX versus normal infrared format. In addition, the sheep races represented in the dataset are marked.

\* Sheep races are: **1.** Norwegian White Sheep, **2.** Norwegian Pelssau and **3.** Old Norwegian Spælsau.

Table 5 shows the number of sheep in the dataset grouped by month and colour. As shown in the table, most of the sheep are white. This uneven colour distribution is representative of the distribution of sheep that can be found in the area where the images were taken. Figure 21 shows a sample of the sheep for each of the respective colours in order to give an idea of how colour classifications were made. Table 5 also shows the distribution of the sheep that are lamb compared to adult sheep. May is the season for lambing so almost 40% of the sheep captured in May are lamb whilst none of the sheep captured in August, September or October are lamb.



**Figure 20:** Definition of extra small (xS), small, medium, large and extra large (xL) sheep. Boxes show the maximum size for the sheep in each classification in relation to the full image dimension of  $3040 \times 4056$ . Images are classified as one of the above size categories by the median diagonal length (D) of all adult sheep bounding boxes. Using the fact that the average adult sheep is 1.3m long, [9], ground sampling distance (GSD) and flight height (H) are estimated for each category by equations 3.1 and 3.2

	Total	Sheep Colour				Sheep Life Stage	
		White	Grey	Black	Brown	Lamb	Adult
August	2890	1643	912	335	0	0	2890
September	1201	727	351	30	93	0	1201
October	1612	860	558	169	25	0	1612
May	2710	1618	169	655	268	1017	1693
All Labelled	8413	4848	1990	1189	386	1017	7396

**Table 5:** Number of sheep in the dataset grouped by month, colour and life stage.



**Figure 21:** A sample of the sheep in the dataset grouped by colour. The exact transition between white, grey and black is somewhat fuzzy.

### RGB Images

The RGB images have a resolution of  $4056 \times 3040$ -pixels. Due to the high relative resolution in the RGB images, sheep are visible from a flight height of up to 100m. One challenge with the RGB images is that there is a lot of reflection and shadows in the case of strong sunlight. Moreover, contrast between the colour of the sheep and their background impacts the sheep visibility. As a result, black sheep are sometimes harder to see due to them looking similar to shadows in the image.

### Infrared Images

The infrared images have a resolution of  $160 \times 120$ -pixels. This is much lower than that of the RGB images. The low resolution makes sheep visible up to between 30-80 m depending on weather conditions. In the case of the infrared images, sheep colour appears less relevant for sheep visibility. Environmental conditions appear important for the visibility of sheep in images since contrast between the sheep and their background is stronger when the ground is cold. In addition, sheep with less wool appear clearer in the infrared images. Another thing to note with the infrared images is a high apparent radial distortion.

The infrared images come in two formats: pure infrared and FLIR MSX (Multi-Spectral Dynamic Imaging), [78]. MSX images are thermal images with embedded edge and outline details from the visible spectrum.

### RGB - Infrared Alignment

The RGB and infrared images are not aligned pixel for pixel. The RGB images cover a larger area on the ground, have a higher resolution and less distortion as compared to the infrared images. As a result, camera calibration and alignment of the images is necessary if information from the two images will be used simultaneously and if there is a desire to have shared labels.

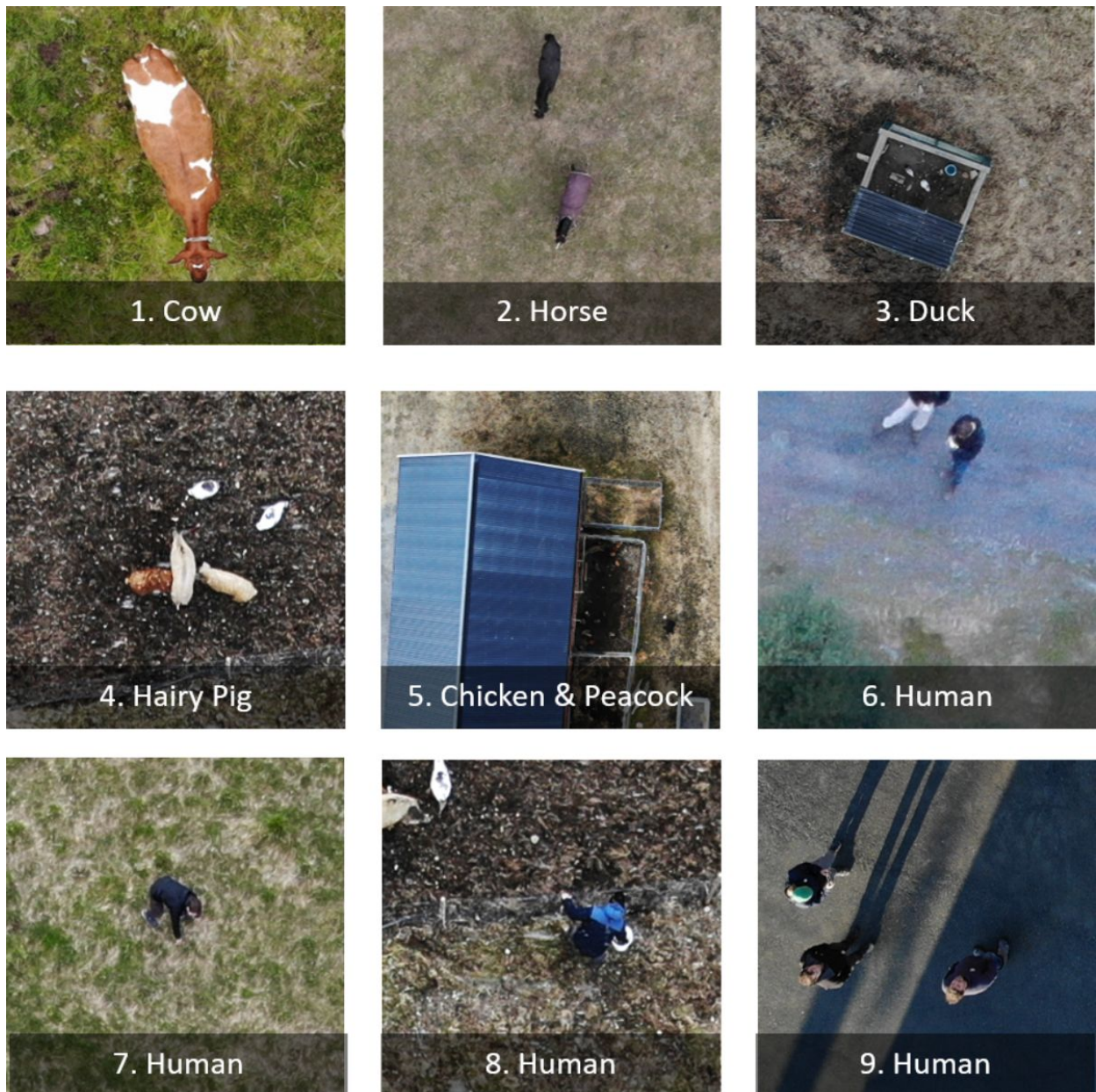
### Seasonal variations

There is seasonal variation in the images taken in the four different months. These variations include variation in temperature, sunlight, vegetation and degree of free ranging sheep. Figure 23 shows some of these variations. Images captured in August are characterised by free ranging sheep, warm temperatures, green grass and dense trees. In contrast, images captured in October and May are characterised by fenced

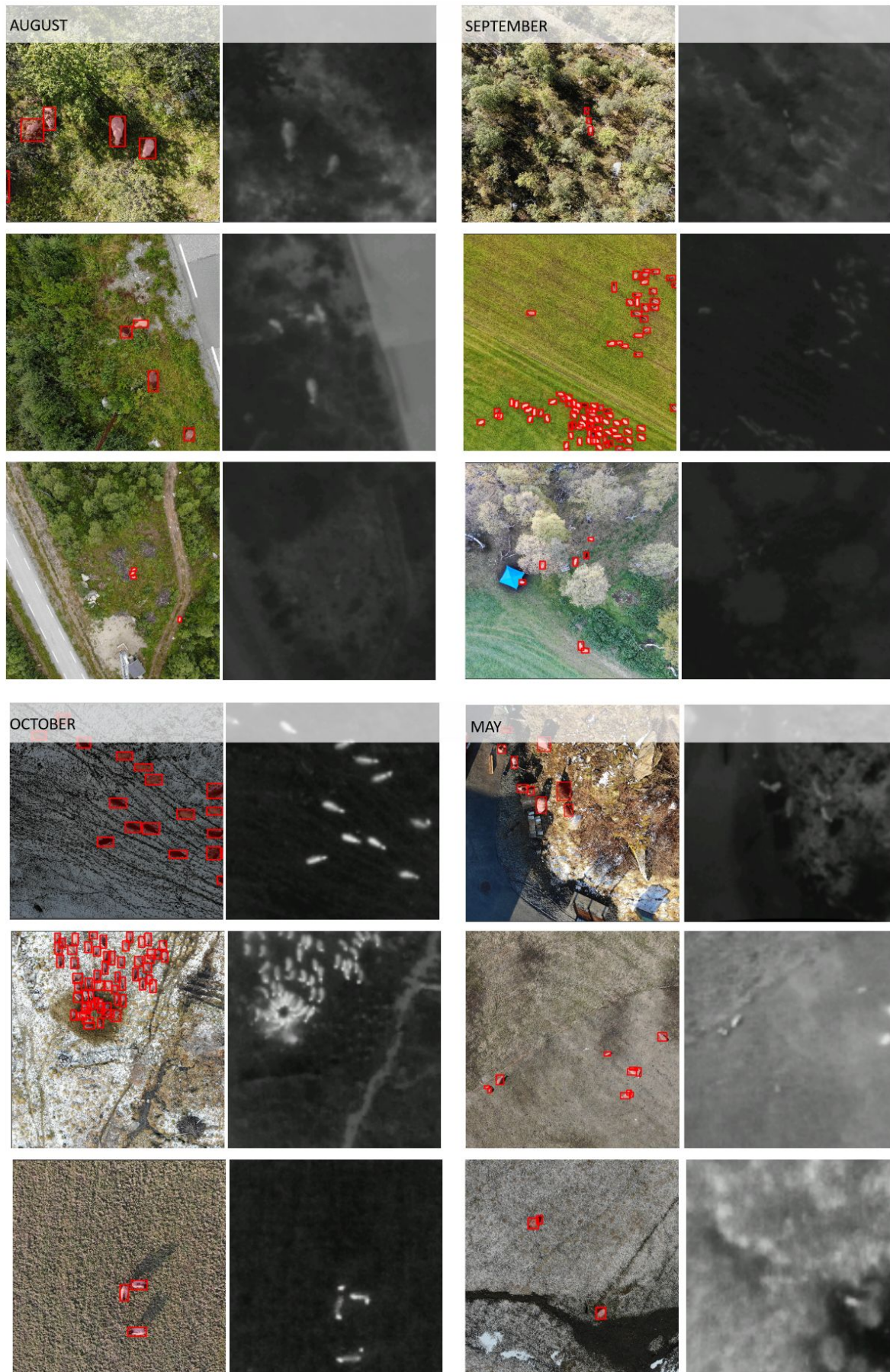
sheep, cold temperatures, snow and leafless trees. This variation is positive as it helps to satisfy requirements R4 and R6 from section 3.2. Another difference is the sheep resolution in the August images is lower since more images were captured from a greater flight height. In addition, the May dataset features a great number of lamb, which is not present in the other months. Another difference between the May dataset and the rest of the images is that the images are taken in a different location and therefore feature another race of sheep. Moreover, the May data differs from the other months because it also has some images that have no sheep in them at all and more images with other animals present.

### Other Animals

Some of the images also contain other animals. A sample of these are shown in Figure 22. Images 1. and 6. in the Figure are captured in August but the remainder of the images in the figures are captured in May.



**Figure 22:** Sample of other animals and humans present in the dataset.



**Figure 23:** Sample of labelled image pairs in the sheep dataset. Labelled RGB image is shown on the left and corresponding infrared image is shown on the right.

### 3.4 Hardware Constraints

The image quality and available computing processing power is limited by available hardware. These hardware constraints come from the specifications of the camera UAV and computer used for training.

#### DJI Mavic 2 Enterprise Dual (M2ED)

The M2ED, [76] (shown in Figure 24) is the UAV used for photographing the sheep. This UAV is a compact and foldable UAV with dual sensors that allow it to capture side-by-side visible and infrared images. More detailed specifications are listed in Table 6. This UAV is chosen for several reasons:

- It is believed that infrared technology can be a great aid in identifying warm, live animals in the infrared relatively colder surroundings.
- It is an affordable UAV. The market price of the M2ED is approximately NOK 30000, which is a price that an average Norwegian farmer would be able to afford.
- It is light and compact, which allows it to easily be transported in challenging terrain.
- It is easy to pilot.

<b>Take-off weight:</b>	899g (without accessories)
<b>Dimensions:</b>	Folded: 214mm x 91mm x 84mm. Unfolded: 322mm x 242mm x 84mm
<b>Max flight time:</b>	31 mins
<b>Max speed:</b>	72kph
<b>Thermal sensor:</b>	FLIR Lepton
<b>Thermal camera resolution</b>	160 x 120
<b>Visual camera resolution</b>	3040 x 4056
<b>Visual camera Field of view (FOV)</b>	Approx. 85°
<b>Visual camera 35 mm format equivalent</b>	24mm

**Table 6:** *DJI Mavic 2 Enterprise Dual (M2ED) specifications.*

A disadvantage of this UAV is the low resolution of the infrared camera, which may limit how high the UAV can fly while still being able to capture sheep in the infrared images. Moreover, flight time for the UAV is limited to 31 minutes. As a result, the UAV is not capable of capturing very large areas in a single flight.



**Figure 24:** *DJI Mavic 2 Enterprise Dual (M2ED).*

#### Computer Hardware

Training is performed on a computer with the following Graphical Processing Units (GPUs) and Central Processing Unit (CPU):

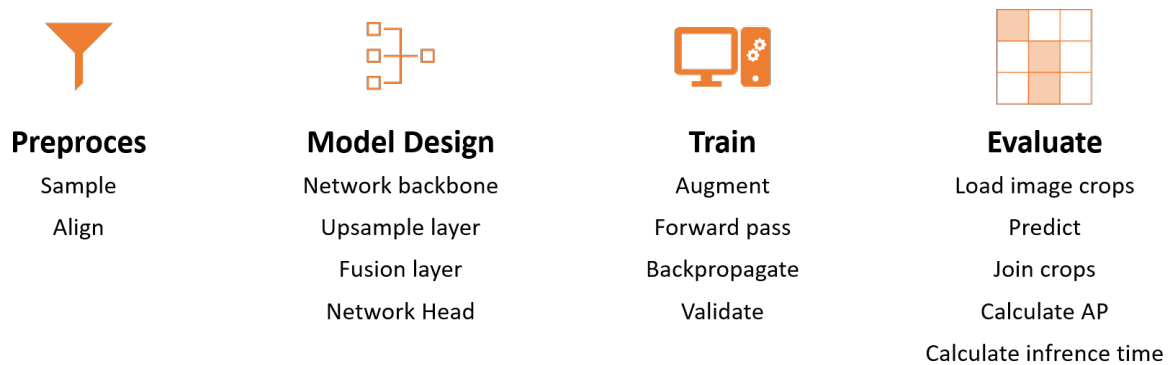
- 2 GeForce GTX 1080, 8GB GPUs
- Intel® Core™ i7-7700K CPU @ 4.20GHz



## 4 Method

This chapter gives an overview of the steps taken to prepare the data, design the model, train the model and finally evaluate the performance of the model. This is all summarised in Figure 25.

Data preprocessing is the first step, which includes image sampling and alignment. Sampling is done by selecting an appropriate sample of the data for further use based on a set of inclusion criteria. Alignment involves finding the parameters to align the infrared and RGB images with each other. Before training can begin, a software environment and CNN model architecture must be designed. The neural network architectures are tailored to suit dual input data and loose localisation quality requirements of the use case at hand. Next, details of the training process are explained and finally an explanation is given for how to use and evaluate a trained model.



**Figure 25:** An overview of the processes performed on the data to prepare, build, utilise and evaluate the sheep detection models.

### 4.1 Data Preprocessing

Some preprocessing procedures are necessary before the data can be used for training and evaluation. There are two main preprocessing steps: image sampling and alignment. Sampling involves selecting which images are to be used further and splitting this data into training and validation sub-datasets. Image alignment involves performing transformations on the infrared images to correct for distortion and to align the image pairs pixel for pixel with one another.

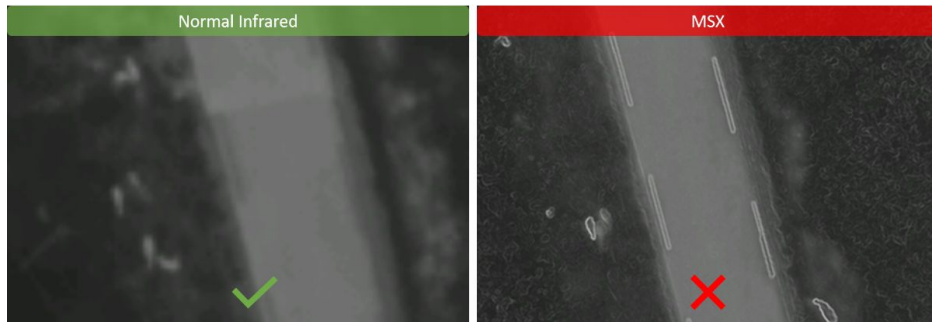
#### 4.1.1 Data Selection and Sampling

A sample of the total collection of images are selected for us in training and evaluating the model. Images are selected based on a set of criteria that are explained in this section.

##### C1: Not MSX

Images where the infrared format is MSX are excluded. MSX images are infrared images that have edge and outline details from the RGB image embedded on them. It is chosen to not include these images because

it is desirable that the sub network responsible for learning features of the infrared image focuses on the information of the infrared image and not on visible spectrum details such as lines. These features will instead be detected by the RGB sub network. Moreover, some misalignment between embedded lines and infrared features are observed in some of the images.



**Figure 26:** MSX infrared images are excluded from use

### C2: Not Extra Small or Extra Large sheep Sizes

The majority of the images in the dataset have small, medium or large median sheep diagonal lengths (as defined in Figure 20) . In the project preceding this thesis, [23] images were not filtered by sheep size and it was found that images with Extra Small sheep were much more difficult to detect. This is likely due to the sheep's low resolution making them less distinguishable from other similar looking items. Therefore, by excluding images with extra small sheep size, it may be possible to reduce the rate of false positive detections. Images with extra large sheep size are also excluded because they are taken from a very low flight height, which is unrealistic for the desired use case. The images that remain will have a diagonal length of between 50 to 300 pixels, which corresponds to a flight height range of approximately 14 to 85 meters.



**Figure 27:** Images with too small or too large sheep are excluded from use. Accepted flight heights correspond to an approximate range of 14 to 85 meters.

### C3: Not Too Similar To Each Other

Since data comes from only 25 separate sessions, variation between sessions is large but variation within one session is often low since the same sheep in the same environment are depicted. Nonetheless, there is still some useful variation within a session from sheep being in different poses and moving around in the environment. Near identical images are removed in order to avoid over-training on certain images.

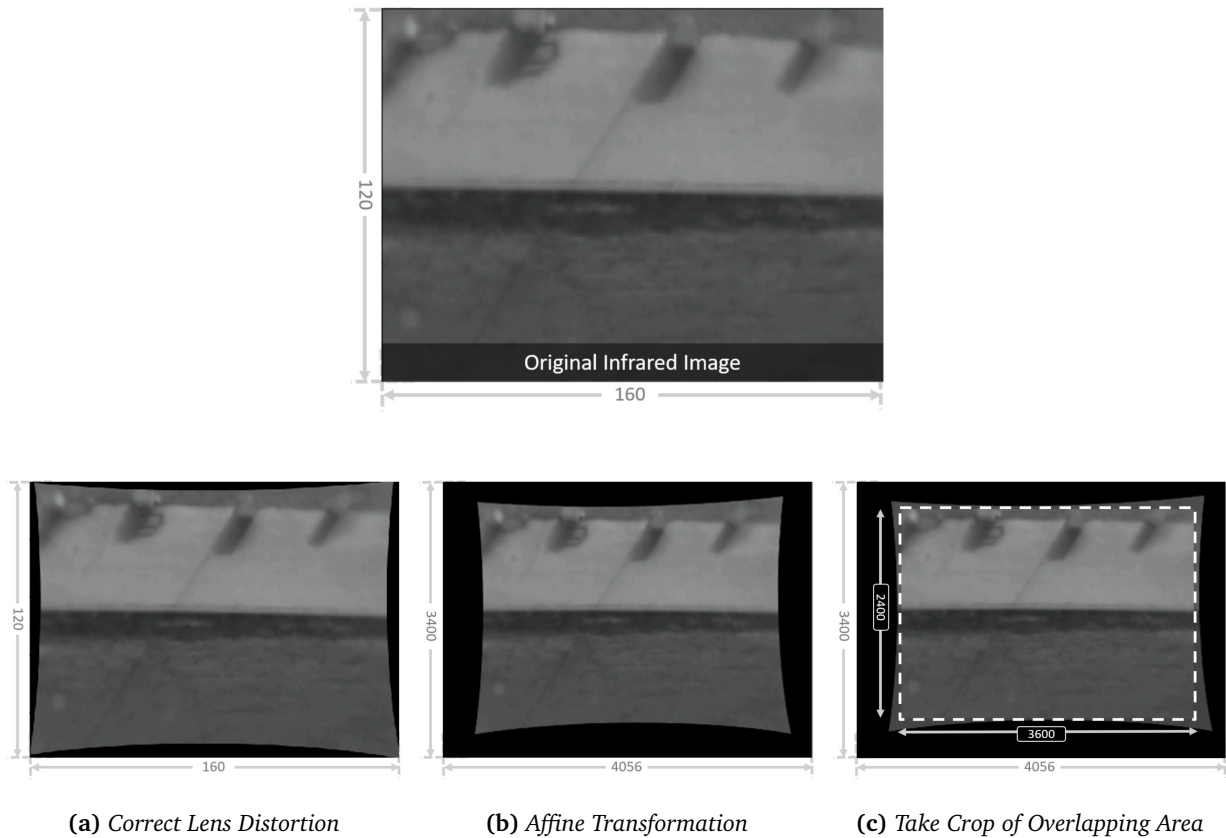
## Training, Validation and Test Split

As mentioned in section 2.4, splitting data into training, validation and test sub-datasets is important for preventing overfitting. In order to assure that these sub-datasets are independent of each other, data from the same session are not used in more than one sub-dataset. Further, it is important that each sub-dataset has enough variation in environment and sheep colour to be representative of the dataset as a whole. In the project preceding this thesis, [23] it was decided to not have a separate test dataset because the dataset is not large enough to create three independent and representative datasets. Instead, the validation dataset was used both for validation and testing. Since sheep in Norway are only grazing outside certain months of the year, it is not possible to collect more data before approximately the middle of May, which is just one month before the thesis deadline. As a result, the available data from August, September and October 2019 is for training and validation. Three sessions with high variation in sheep colour, sheep size and grazing environment are selected from the twenty sessions in 2019 to use as a validation dataset. The remaining seventeen are used for the training dataset. The new data collected in May 2020 was intended to be used as a test dataset, however it differs too much from the training dataset that it would be unreasonable to judge the final model performance on this data. Instead, the May 2020 data is used as case studies to assess how the model generalises to lamb and unseen sheep races.

### 4.1.2 Camera Calibration and Image Alignment

Since there is a misalignment between the infrared images and the corresponding RGB images in the dataset, it is necessary to perform camera calibration to align the images with one another. The misalignment comes from a radial distortion present in the infrared images as well as the infrared camera having a lower resolution and a smaller ground coverage area for the same flight height. By transforming the infrared images to the RGB image's camera coordinate system, it is possible to use the same labels for both images, which is beneficial when the goal is to use both images simultaneously to predict the same output.

Image alignment is solved with a three step process, which is depicted in Figure 28. First, the infrared images are corrected for lens distortion using camera calibration. Specifically, OpenCv's [70] implementation of Zhang's 'Flexible New Technique for Camera Calibration, [73] is used. Secondly, the affine transformation, which maps pixels of the undistorted infrared images (output from step1) to their corresponding pixels in the RGB image is calculated using the least squares method on a series of corresponding (undistorted) infrared and RGB image points. Finally, a crop of the overlapping area is taken from both the RGB and infrared images and appropriate adjustments are made to the labels. The result should be a  $3600 \times 2400$  resolution aligned and labelled RGB and infrared image pair.

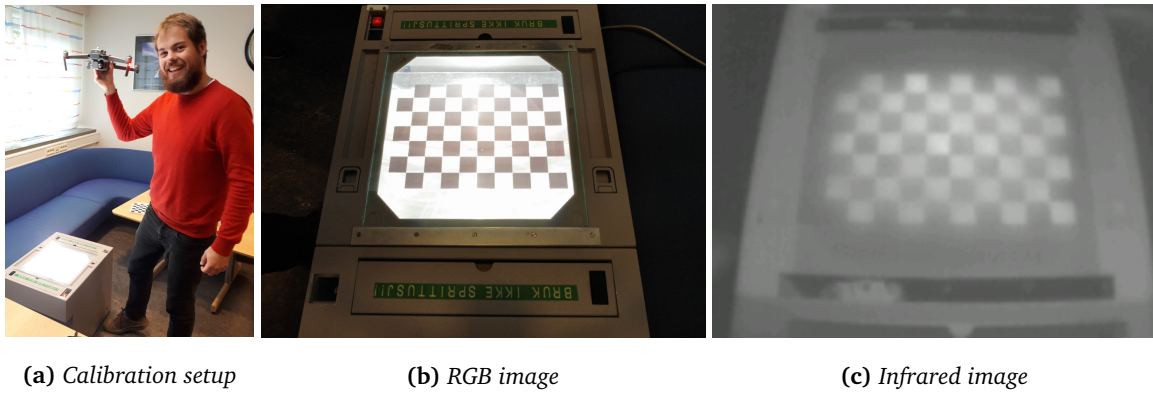


**Figure 28:** Three step process for aligning the infrared and RGB images

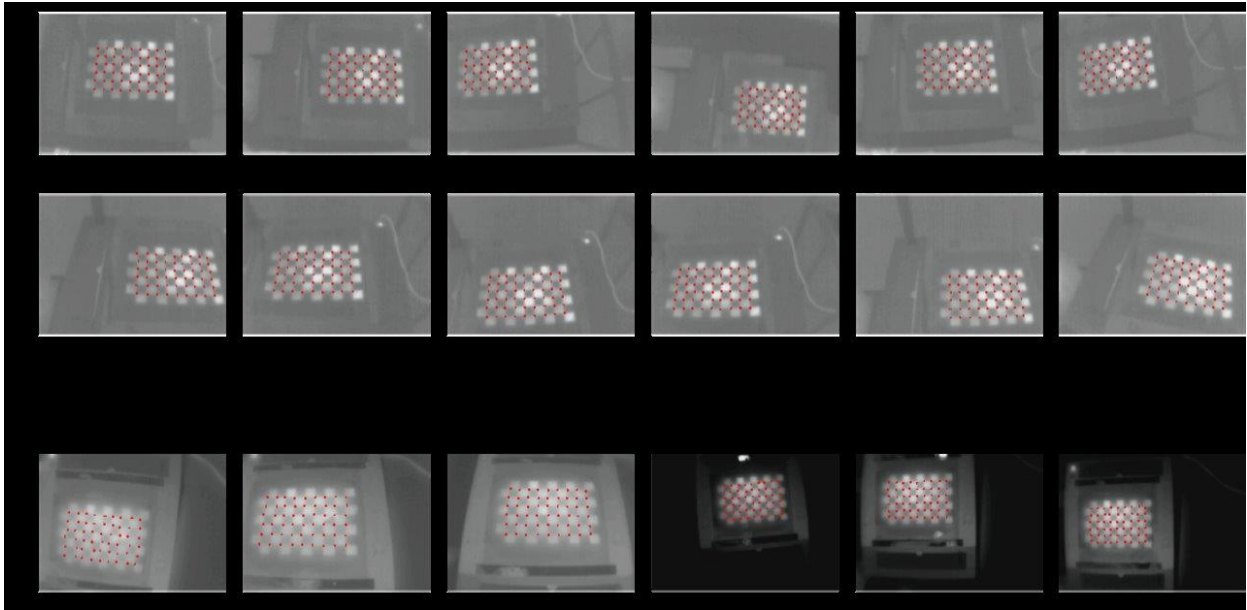
### Correcting Lens Distortion

OpenCV's tutorial on camera calibration, [70], is used to correct the distortion present in the infrared images. The method is based on Zhang's approach as described in section 2.6. The tutorial suggests to use a chess board pattern as the planar model since it is very well defined. The corners of the chess board are used as the image points  $(x, y)$  and a simple grid is used for corresponding the model object points  $(X, Y)$ . Given multiple samples of image points from this planar chess board pattern, the openCV function `calibrateCamera()`, [70] can be used to estimate the camera matrix and distortion coefficients. These coefficients can be saved and later used to undistort any image taken using this camera.

Since the camera to be calibrated is an infrared camera, the chessboard pattern must be visible in the infrared spectrum. An overhead projector and transparency film with the printed chessboard pattern are used for this purpose. This will make the pattern visible in the infrared spectrum because the projector will heat up the black squares of the film but not the transparent ones and as a result, the temperature difference will be visible in the infrared image. Figure 29 shows the setup for the camera calibration process. The low resolution and blurry nature of the corners in the infrared images makes it difficult to automatically detect the chess board corners so this is done manually. To compensate for the inaccuracy caused by the blurry nature of the feature points and human error, a great number of images (72) are used. Figure 30 shows a sub sample of the checker-board photos with marked corner points that were used for camera calibration.



**Figure 29:** Setup for camera calibration showing the overhead projector, the dual imaging UAV and a sample of the RGB and infrared image pairs taken by the camera.

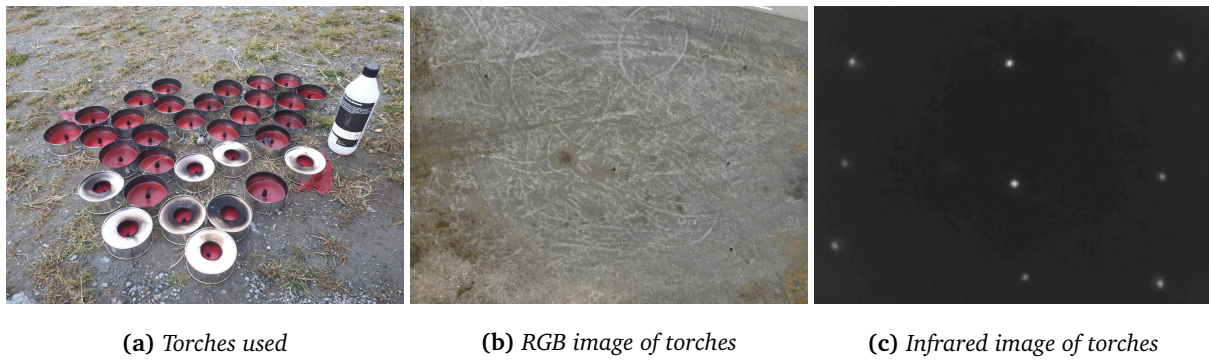


**Figure 30:** Sub-sample of the in total 72 checker-board photos with marked corner points used for camera calibration

### Affine Transformation from Infrared to RGB Coordinate System

Once the infrared images have been undistorted by the camera calibration process, the transformation between the undistorted infrared coordinate system and the RGB coordinate system (assumed distortion free) should be an Affine transformation ( $T$ ). The relationship between the two coordinate systems can therefore be described by equation 2.8. The least square method can be used on this relationship to estimate  $T$  given a set of corresponding points in the two coordinate systems.

The python library scikit-image's *transforms* module [79] provides the method `estimate_transform()`, which is used to solve the least squares problem and estimate  $T$ . Outdoor torches are used as marker points because they are visible in both the RGB and infrared images. Figure 31 shows the torches used and how they appear in the RGB and infrared images.



**Figure 31:** Images of torches, visible in both visible and infrared images

### Cropping of Overlapping Area

The final alignment preprocessing step is to extract crops of  $3600 \times 2400$  from the aligned infrared image and the corresponding RGB image. This is done so that all areas of the image contain both RGB and infrared data. In addition, the bounding boxes need to be translated and trimmed to fit the new cropped image.

## 4.2 Software Environment

PyTorch 1.3.0 for python 3.7.4 and CUDA 10 is the chosen software environment. PyTorch, [80] is a popular open source machine learning framework that enables powerful tensor computations on Graphical Processing Units (GPU). PyTorch is mainly developed by the Facebook AI research lab and has a python interface. PyTorch was chosen based on personal experience. Moreover, it is among the most popular frameworks in research, [81], it is simple to get started with, it comes with some pretrained models. Starter code from the deep learning course TDT4265 at NTNU, [82] is used as starter code.

## 4.3 Neural Network Architectural Design Choices

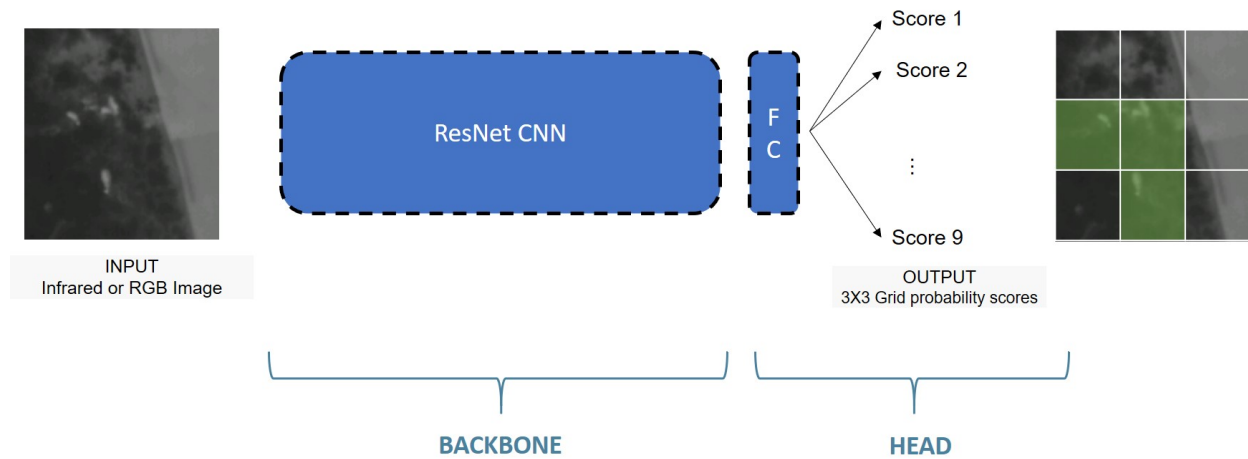
This section describes the neural network architectural choices made to design the sheep detection model. Choices are based on the theory and related work discussed in chapter 2, the system requirements explained in section 3.2 as well as the software and hardware constraints presented in sections 4.2 and 3.4 respectively.

Two separate network architectures are used. One network accepts a single input (either RGB or infrared), whilst the other network accepts both RGB and infrared inputs simultaneously. The single-input network is shown in Figure 32 and consists of two parts: The backbone network and a network head. This design is in fact identical to a simple classification network with 9 classes, however instead of the output representing the probability scores for 9 classes, the output represents the probability of sheep being present in 9 regions of the image.

The fusion network shown in Figure 35 shows how information from *both* the RGB and the infrared image is used to predict a single output. This design is based on the network designs used in similar multi-channel computer vision solutions presented in section 2.5.

### 4.3.1 Backbone Neural Network

As explained in section 2.4, the backbone neural network is responsible for computing features of the input image. Various variants of ResNet and ResNeXt are tested as the backbone because these networks achieve SoTA results, are easy to scale and the PyTorch framework provides pretrained weights for them that are simple to begin using.



**Figure 32:** Network architecture used for RGB only or infrared only input

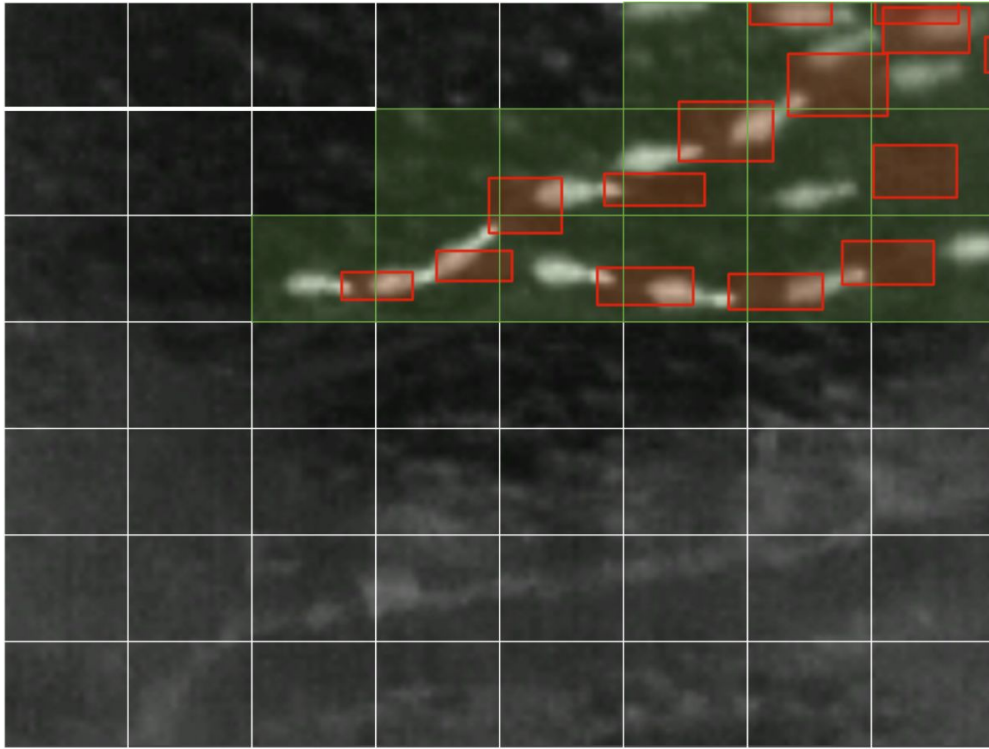
### 4.3.2 Network Head

As discussed in greater detail in section 2.4, the network head is responsible for producing the desired output from the feature map. In typical object detection tasks, the desired output is a set of bounding boxes for each of the classes. Accurate bounding boxes are crucially important to tasks such as automating a robot arm or a self-driving car. However, as stated by requirement R1 (section 3.2), such precise localisation quality is not necessary for the use case of detecting sheep. For the desired use case it is sufficient for the model to simply say whether a certain area has sheep present or not. Moreover, it is not necessary to find each individual sheep in a group of sheep as long as at least one positive sheep detection is made in that area. As a result, this greatly simplifies the problem to be solved. Instead of the network outputting bounding boxes for each individual sheep, a fixed size output is computed, where each number in the output represents a confidence score for sheep presence within a certain area of the image. Figure 33 shows the desired grid output for an input infrared image. The 7x8 grid output is created predicting probabilities of 3x3 grids on crops of the image and concatenating the result.

From observing Figure 33, another argument for loosening the localisation quality demands is made clear. The misalignment present between the bounding boxes transformed from the RGB image onto the infrared image is grossly misaligned when the sheep in the image are moving in their environment. As a result, the same bounding box outputs from the RGB and infrared inputs cannot be expected. However, despite bounding box misalignment, the grid cells with sheep present will still be mostly correct. It should be noted the misalignment present in Figure 33 is on the extreme end as the sheep are moving fast. Most of the projected bounding boxes have a much better alignment.

This grid output is inspired by the object detection architecture YOLO v1. In YOLO v1, the input image is divided into a 7x7 grid, where each cell is responsible for predicting 2 bounding boxes and class confidence scores for each of the classes in the dataset. For the pascal VOC dataset, this results in an output size of  $7 \times 7 \times 30$ . The main disadvantage of YOLO v1 is that each grid can only predict one class, which means that detecting many small objects very close to each other becomes an issue. However, this is not a problem for the desired use case as it is satisfactory to find just one sheep in each cell. The conceptual comparison between outputs of YOLO v1 and the sheep network is shown in Figure 34. By using this type of YOLO inspired output, the network is able to inherit the advantages of speed, low false positive rate and ability to exploit contextual information that YOLO v1 provides without necessarily being negatively impacted by disadvantages of YOLO v1.

This simplification should both improve detection performance and reduce inference time. Improved performance is expected because the output is much simpler, and detection of individual closely clustered sheep is not necessary. A faster inference time is expected since no complex bounding box computations are necessary.

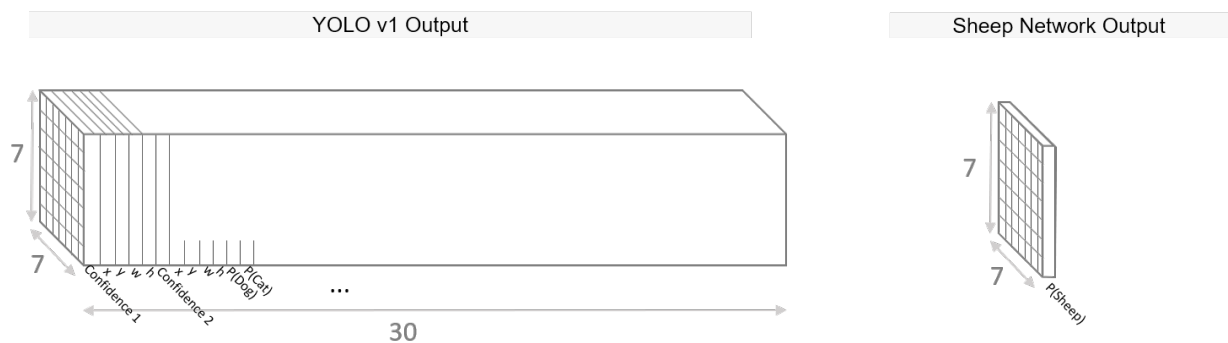


**Figure 33:** Desired grid result when an infrared image is processed by model. Red boxes are bounding boxes that have been transformed from the RGB coordinate system and green-shaded grid cell represent cells where the detector has made a positive sheep detection. As shown in the image, the transformed bounding boxes do not align well with the sheep in the image since the sheep are moving in their environment, however the grid classifications are still mostly correct

### 4.3.3 Fusion Network

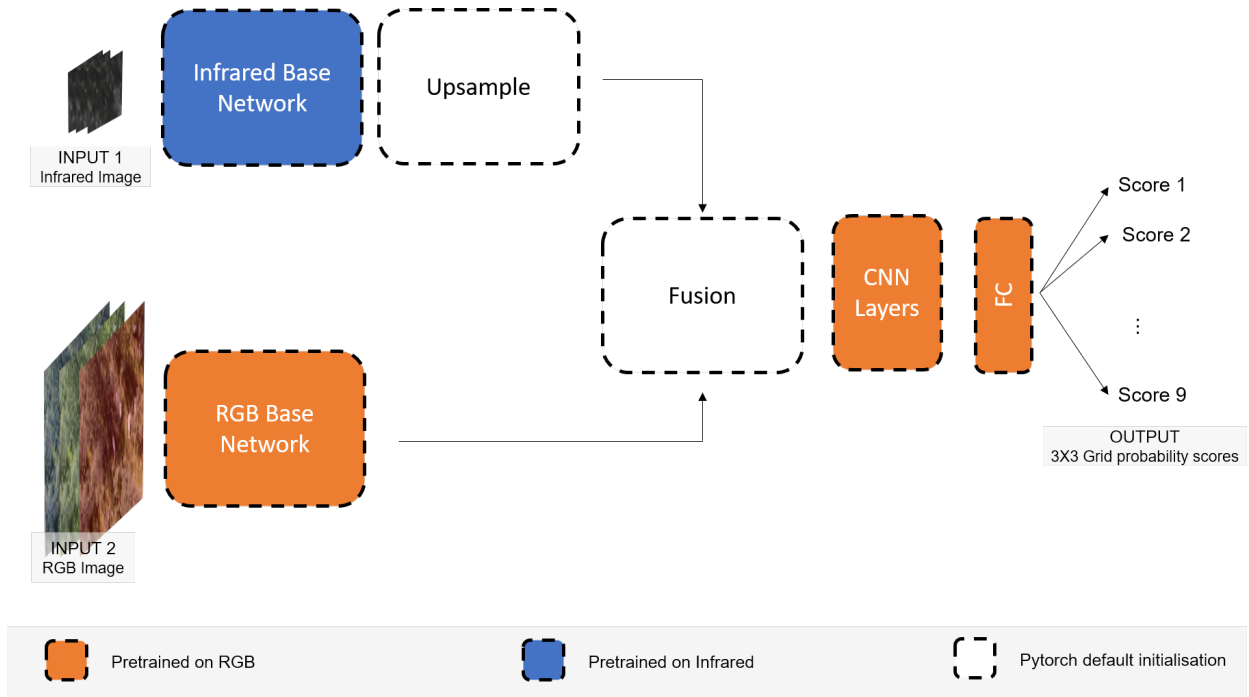
The fusion network shown in Figure 35 is used to compute useful features from both RGB and infrared input. This network architecture is based on the architecture described in [6] and [7], where the two separate inputs are processed in two parallel subnetworks, who's outputs are fused together somewhere in the middle of the network to produce a joint information feature map.

In order to give the network a warm start, the fusion model is initialised with weights that have been trained on infrared and RGB data separately. Fusion is attempted at various mid to late positions of the ResNet architectures since previous research found mid to late fusion to be the most effective. Figure 36 shows the various depths of ResNet where fusion is tested. Two key components of the fusion model are the up-sample layer and the fusion layer. The up-sample layer is used to increase the spatial resolution of the infrared feature map to match that of the RGB feature map and the fusion layer is responsible for joining the feature maps of the two sub networks.

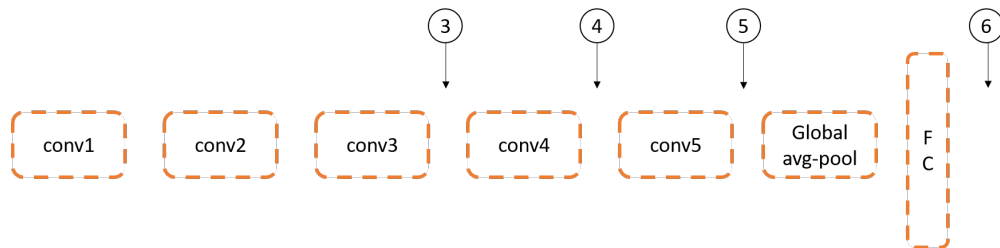


**Figure 34:** Output of YOLO v1 on the pascal VOC dataset compared to the conceptual grid output of the sheep dataset.





**Figure 35:** Fusion Network. This architecture is used for accepting both RGB and infrared input



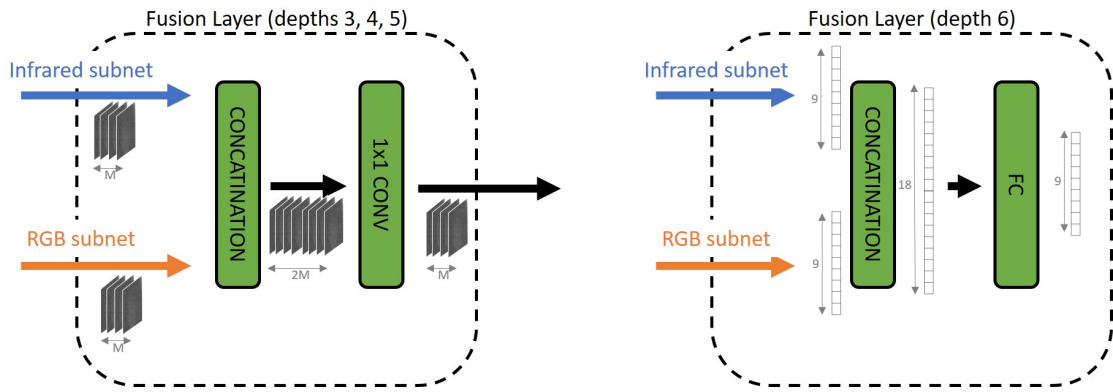
**Figure 36:** The various depths of ResNet where fusion is attempted. ResNet layer names and detailed architectures are described in Table 2. Fusion at depths 3,4 and 5 is done by concatenation and  $1 \times 1$  convolutions, whereas fusion at depth 6 is done with concatenation followed by an extra fully connected layer

### Up-sample Layer

Since the infrared images have a much lower spatial resolution than the RGB images, upsampling of the infrared feature map or down-sampling of the RGB feature map is necessary so that the spatial resolution of the two feature maps will match prior to fusion. Upsampling on the infrared feature map is done rather than down-sampling of the RGB feature map in order to not lose any information. Transposed convolutions with a kernel size of 3, stride 2 and padding 1 are used to double the spatial resolution of the infrared image. As a result, the input RGB resolution divided by the infrared resolution should be a factor of  $2^x$ . For example, if the RGB input image resolution is 1024 and the infrared input resolution is 64 then the infrared feature map resolution needs to be doubled 4 times to match the resolution of the RGB feature map. This is done by performing 4 transposed convolution operations. Performing upsampling by transposed convolutions may be advantageous because the network will learn the best way to upsample rather than relying on predefined interpolation methods.

### Fusion Layer

The fusion layer architecture can be seen in Figure 37. The fusion layer is responsible for joining the feature maps from the two sub networks to a single feature map. Due to the different shape of the inputs, fusion at depths 3,4 and 5 is done by concatenation and  $1 \times 1$  convolutions, whereas fusion at depth 6 is done with concatenation followed by an extra fully connected layer. Convolutions are used to reduce the number

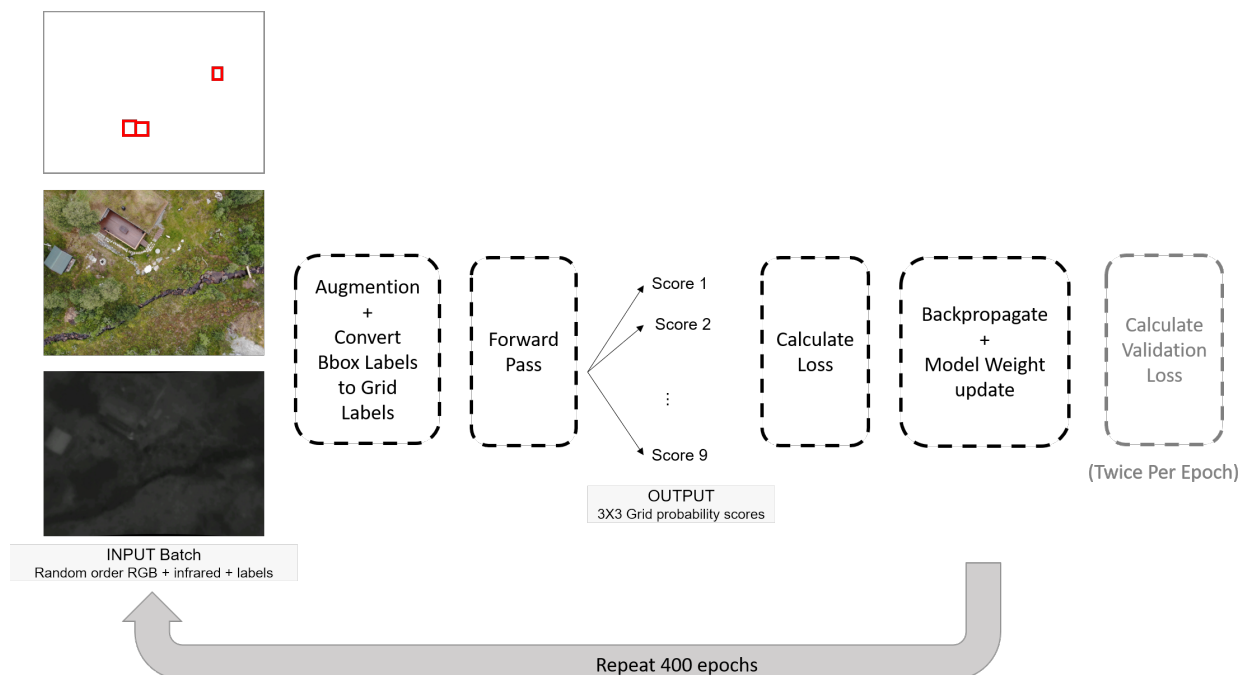


**Figure 37:** Fusion layer architecture. Fusion is done by concatenation followed by either  $1 \times 1$  convolutions if fusion is performed mid-ways in the network or an extra fully connected layer if fusion is performed after the fully connected layer.

of layers to match the expected number of input layers in the next ResNet layer. By performing fusion by convolution, it is hypothesised that the network will learn to select the most useful elements of the two feature maps.

## 4.4 Training

Training is done to specialize a model at detecting sheep in images. Figure 38 shows the steps involved in training the CNNs. Images are selected in random order from the training dataset. First, augmentation is performed, and the bounding box labels are translated to grid labels. Next the images are processed by the CNN to produce suggested classifications for 9 grid cells. Loss is calculated by comparing the network output to the ground truth labels and weights are updated accordingly by backpropagation and gradient descent. This process is repeated until the network has 'seen' every image in the training dataset 400 times (400 epochs). In addition, the training process is monitored by calculating the loss on the validation dataset twice per epoch.



**Figure 38:** The Training Process

## Data Augmentation

As mentioned in section 2.4.4, augmentation is a useful tool for increasing the variation in the dataset and avoiding overfitting especially when the dataset is small. Figure 39 shows how augmentation is performed on the images. The Albumentations library, [83] is used to implement the augmentations as it easily allows for fast and flexible simultaneous image and label augmentation. Further details about specifics of the augmentation and justification for use is found in Table 7



**Figure 39:** Training Augmentation Pipeline

Augmentation	Probability	Details and justification
Flip	0.5	The Image is flipped vertically and/or horizontally with a 50% probability. Since the photograph is a UAV image, any degree of flipping and rotation would be realistic.
Rotate	0.5	Since there is no up or down in the UAV images, the images can be rotated randomly anywhere from 0 to 360 degrees.
RandomSizedCrop	0.5	A random square crop is taken somewhere on the input image. The lengths of the crop edges are somewhere in range [1000, 1400]. By taking a random crop somewhere on the image, the input will be significantly different each time and the network will not be able to learn unwanted features such as sheep often being present in the centre of the image. Moreover, a great number of the images will not contain sheep at all which is also useful for the network to train on. By ranging the crop size, the network will also be conditioned on a greater range of sheep sizes that will naturally occur from variances in terrain height.
Resize	1.0	The RGB images are resized to a fixed square size depending on which resolution is being tested. For RGB images, resolutions ranging from 128 to 1280 are tested. The infrared image is resized to a smaller size because the original image resolution is low. Due to the NN design, the size of the infrared image should be a multiple of 32, at least 64 pixels and the RGB dimension should be divisible by the infrared dimension by a factor of $2^x$ , where x is an integer.
OneOf: <ul style="list-style-type: none"> <li>• RandomContrast</li> <li>• RandomGamma</li> <li>• RandomBrightness</li> </ul>	0.5	This augmentation is applied only to the RGB images. One of the three colour augmentations is applied to the RGB image with a 50% probability. Default limits of the albumentations library are used. Altering the colour properties of the RGB image will expose the network to a greater range of possible colour properties in the images that may occur under varying weather conditions.
Normalize	1.0	Normalising the images is done by subtracting the mean per channel and dividing by the standard deviation per channel. This is done to centre the data around zero and give the images a similar range. For the RGB images, the default mean and standard deviation is used. However, for the infrared images, a mean value of 0.2 for all channels is chosen based on the properties of the images.

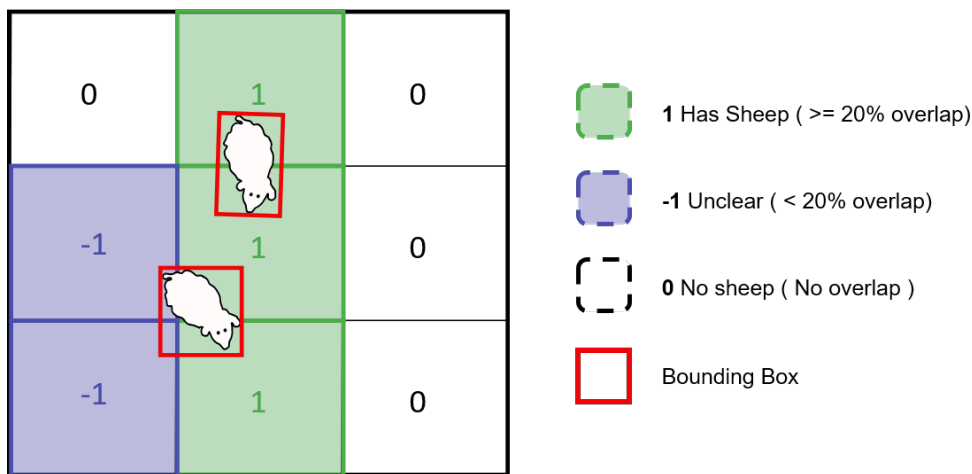
**Table 7:** *Explanation of Chosen Augmentations*

## Converting Bounding Box Labels to Grid Labels

Since the output of the network is 9 grid labels, it is also necessary to have ground truth labels in this format. These ground truth labels are created using bounding boxes that have undergone all the geometric augmentations described in Table 7. It is chosen to classify each grid cell of the cropped image as one of three categories:

1. **Has Sheep:** The grid cell has sheep if any bounding box exists with at least 20% of the bounding box area overlapping the grid cell.
- 1. **Unclear:** If there exists overlap with one or more bounding boxes but none of the overlapping bounding boxes overlap by more than 20% of the bounding box area, it is unclear if the grid should be classified as having sheep or not. As a result of the fuzzy nature of these cells, these cells are marked as unclear and the network output for these cells will not contribute to the total loss.
0. **No Sheep:** If there does not exist any bounding boxes that overlap with the grid cell, the cell is labelled as not having any sheep.

An example of grid classification on a crop with two bounding boxes is shown in Figure 40



**Figure 40:** How ground truth grids are defined from bounding boxes

## Loss

Loss is calculated by first removing predictions and ground truth values for unclear grid cells. Then, PyTorch's MultiLabelSoftMarginLoss is used on the remaining values. This is a type of log loss, which is based on max entropy between input and target. The multi-label loss is suitable because the actual output of the network is identical to a classification network with 9 possible classes, however more than one of the labels can be true.

## Monitoring Training Progress

In order to monitor the training process during training, both training loss and validation loss are logged twice each epoch. This is useful for fine tuning the model parameters and for knowing when the model is beginning to overfit the training data. For example, if the training loss progress is very volatile, this can be a sign that learning rate is too high. On the other hand, if the training loss plateaus at a higher than expected value then it could mean that the learning rate is too low. When training loss continues to decrease but validation loss has stagnated or begun to increase, this is a sign of the network overfitting to the training data.

## 4.5 Experimental Variables

The goal of this thesis is to develop a fast, precise and sensitive deep learning model for detecting sheep in RGB and infrared UAV images. Performance of the model is dependent on the action and interaction of a great number of variables that must be optimised or systematically analysed to determine their best values. The performance metrics are the dependent variables that determine how good or bad the model performance is. The independent variables are those variables that are assessed by the research questions. These variables are systematically altered in order to determine their impact on model performance. Finally, the control variables are those variables that must be decided but whose impact is not extensively analysed. Some of the control variables such as general neural network architecture and general training procedure have already been presented earlier in this chapter.

### 4.5.1 Performance Metrics

There are two main performance metrics that are assessed. These are average precision and inference time. Achieving a high average precision is important for the trustworthiness and usability of the detection model, assuring that as many sheep as possible are found with as few false positives as possible. However, a low inference time is also important for the model to be usable in the imagined use cases. Unfortunately, measures to improve one of these performance variables often has detrimental effects on the other. As a result, the main goal is to discover a set of models that achieve an acceptable trade-off between the two performance metrics.

#### Average Precision

AP is described in section 2.3.1. Since the output of the models is grid probabilities rather than bounding boxes, it is not necessary to calculate IoU in order to determine whether a detection is a true or false positive. Average precision is a performance metric that takes both precision and recall into account. Precision is a metric that describes the proportion of the sheep detections made that are true detections. However, precision does not consider all the detections that were missed (false negatives). These missed detections are considered by recall, which describes the proportion of all the grid cells containing sheep that were found by the detector.

#### Inference Time

Inference time is defined in this thesis as the time in seconds that it takes to predict the  $7 \times 8$  grid output of a pre-aligned aligned image pair using the available hardware described in section 3.4. The main processes considered by inference time calculations are shown in Figure 41. Processes are occurring simultaneously on the machine's GPU and CPU. The scope of this project is limited to analysing the CNN model performances. As a result, processing time for image alignment is not analysed.

As mentioned in section 4.3, The  $7 \times 8$  grid output is created by predicting probabilities of  $3 \times 3$  grids on crops of the image and concatenating the result. Inference time is calculated by running inference on the training dataset and dividing the total time taken by the number of images in the dataset. Since inference time can vary slightly due to other background processes happening simultaneously on the machine, this process is repeated twice, and an average is used.



Figure 41: The processes considered by inference time

#### 4.5.2 Independent Variables

The independent variables are the variables investigated in the research questions mentioned in section 1.2. The justification for choice of independent variables and the levels that are investigated are explained below.

##### Input Image Type

Three input image types are considered:

1. RGB
2. Infrared
3. RGB + Infrared (RGB+I)

RGB images contain useful colour and texture information that the network could use for detecting sheep, however weather conditions and sheep colour will affect the performance of the model. Infrared images contain useful temperature information that is not present in the RGB image. However, the low resolution of the infrared images limits the amount of useful information and it may be challenging for the model to distinguish sheep from other warm objects. Finally, using both the RGB and infrared images as input is expected to give the best average precision performance because the model will be able to take advantage of the colour and texture information from the RGB image as well as the temperature clues from the infrared image.

On the other hand, inference time will likely be highest for the fusion model that takes RGB+I input because the model must perform more operations to calculate the output. Due to the low resolution of the infrared images, the infrared only models are expected to have the best inference time.

##### Image Resolution

Image resolution refers to how much down-sampling of the original image is done before passing the image through the CNN. Reduced inference time is the motivation for performing this downsampling, however it may have undesirable consequences on the average precision result due to information lost in the downsampling process. Since the infrared images already have a very low resolution compared to the RGB image, the effect of image resolution is only investigated with regards to the RGB images. The goal of varying RGB resolution is to find an acceptable degree of downsampling that is able to maintain a desirable average precision result.

The images are classified as 1200 pixel wide square crops. For the RGB only model, the following resize shapes are attempted: 1280, 1152, 1024, 896, 768, 640, 512, 384, 320, 256, 160 and 128. This tests a range of 10 to 100 % resolution. Each resolution is tested twice.

Due to time limitations, not all the above resolutions are tested for the fusion model (RGB+I). The following RGB sizes are tested once for all fusion depth levels: 1024, 768, 640 and 512. In addition, sizes of 256 and 128 are tested for the best performing fusion depth level.

## Network Depth and Cardinality

As mentioned in section 4.3, ResNet and ResNeXt are chosen as backbone networks. As explained in section 2.4, ResNet exists with a range of different depths. The deeper variants have better performance but also take more memory and longer time. ResNeXt comes in the same depths as ResNet but with the extra dimension of cardinality, which refers to the number of parallel transforms in each block. Previous research showed that adding cardinality to the network improves performance, [45].

Network depths of 18 and 50 are tested for the three input types. In addition, a network depth of 101 is tested for the infrared only network. The 101-layer network is not tested for the RGB only and RGB+I networks due to insufficient GPU memory for the desired image resolutions. In addition, ResNeXt 50 with cardinality of 32 is tested for all three input image types.

## Fusion Depth

Fusion depth refers to where in the CNN backbone architecture fusion is performed. Previous research (see section 2.5) found mid to late fusion to perform the best with some variation by class. As a result, fusion is tested at four different mid to late locations in the in the network as shown in Figure 36.

### 4.5.3 Control Variables

Control variables are those variables that are kept constant or controlled for all experiments. Some of the control variables related to network design and training have already been presented earlier in this chapter, however they will be listed here as well. This section explains the most important control variables and how they are set.

## Learning Rate

As mentioned in section 2.4, choosing a suitable learning rate is crucial for a successful training of a CNN. Therefore, time should be spent to optimise this parameter in advance of running tests. A new learning rate is found for each of the three networks by a grid search. This is done by observing the progression of training loss for 50 epochs for a range of learning rates. The learning rate that plateaus or trends towards the lowest loss is selected. Once the learning rate is found, the same learning rate is used for all experiments with linear scaling in relation to batch size. Table 8 shows the learning rate and batch size values that are used.

## Optimiser

The Adam optimiser is chosen over stochastic gradient descent because the Adam optimiser causes faster convergence.

## Batch Size

Similarly to learning rate, a suitable batch size is found by doing a grid search for a range of batch sizes (within the limits of what fits on the GPU memory). As suggested by literature, a linear scaling of learning rate is used when adjusting batch size.

	Learning Rate	Batch Size
<b>Infrared</b>	0.0001	32
<b>RGB</b>	0.00007	8
<b>Fusion (RGB + I)</b>	0.00005	8

**Table 8:** The learning rate and batch size used for experiments



## Number of Epochs

It is chosen to run training for 400 epochs because this appears to be enough time for convergence to happen and for the model to begin overfitting.

## Output Grid Shape

An output grid shape of  $7 \times 8$  is chosen somewhat arbitrarily based on the inspiration from the  $7 \times 7$  YOLO v1 architecture. This grid size appears coarse enough to be a simplification of the problem but also small enough for a sheep in the grid to represent one of the main defining features of the grid cell.

## Augmentations

Augmentations are performed on the image during training to increase the variability of the dataset and thus avoid overfitting. The exact augmentations and their values are outlined in Table 7.

## Number of Workers for Data Loading

The number of workers for data loading refers to the number of processes assigned to generating data batches in parallel on CPU. This is an important variable for minimising the time used for training and inference. Ideally, the next batch of data should be ready by the time the CNN on the GPU is done processing the current batch so that data loading is not bottle necking the processing time. As with learning rate and batch size, this number is found by trial and error and is fine tuned in order to maximise inference time. By testing various number of workers, for different configurations, it was found that 8 workers was best for image resize sizes lower than 160 pixels, 16 workers for image resize sizes between 160 and 320 pixels and 24 workers for image sizes larger than 320 pixels. No further gain to inference time was obtained by increasing the number of workers to more than 24.

## 4.6 Inference and Evaluation

The following steps are taken to predict grid values for a dataset and evaluate the result:

1. Load the CNN model to be tested.
2. Start the timer.
3. Load image data as  $1200 \times 1200$ -pixel crops. At test time, no geometric or colour augmentations are applied (except normalisation).
4. Predict  $3 \times 3$  grid confidence scores on the  $1200 \times 1200$ -pixel crops by running the model on the images.
5. Join the results on the cropped images to get  $7 \times 8$  grid predictions for the full image. In the case of overlapping grid predictions, use an average.
6. Stop the timer.
7. Calculate inference time by dividing the time recorded by the timer by the number of images in the dataset.
8. Calculate average precision using the python library sklearn's method `average_precision_score()`, [84]. Unclear cases as defined in Figure 40 should be excluded from the calculation since it is unclear what the ground truth value of these cells should be.

## 5 Results

The results chapter is divided into two main parts: preprocessing results and experimental results. The preprocessing results section presents outcomes from following the data preparation steps outlined in section 4.1.1. The experimental results section presents average precision and inference time results for the various model configurations and provides the basis for answering the research questions of the thesis. Model performance is shown in relation to input datatype (RGB, infrared or RGB+I), network design (ResNet18, ResNet50 and ResNeXt50 with various fusion levels) and RGB resolution. Furthermore, model performance with regards to various qualities of the images (median sheep size, sheep colour and sheep life stage) are presented for the highest average precision model of each input type in order to identify the models' strengths and weaknesses.

### 5.1 Data Preprocessing

Data preprocessing involves two main processes: image sampling and image alignment. This section presents results of the two processes. The sampling results include an overview of the distribution of the post-sample data with regards to sheep grazing environment, median adult sheep size, sheep race, sheep colour and sheep life stage. The alignment results present the resulting calibration parameters from following steps outlined in section 4.1.2 and gives some visual and numerical examples of images and points that have undergone the alignment process.

#### 5.1.1 Data selection and sampling

After sampling by the criteria explained in section 4.1.1, 515 images remain. These images are further divided into four sub-datasets: training, validation, T1 and T2. Some samples of images from each of these four datasets can be seen in appendixes A.2, A.3, A.4 and A.5. The training and validation data is taken from August, September and October of 2019 since the May 2020 data was not yet collected at the time when the networks were being trained. The images collected in May 2020 (datasets T1 and T2) are unfortunately not appropriate for a test dataset because of reasons explained in section 4.1.1. Instead, these datasets are used as case studies to assess the models' generalisation ability towards lamb and a new sheep race. Data from T1 is captured in an area called Klæbu and has an abundance of lamb and adult sheep in the familiar Norwegian White Sheep race. In comparison, data from T2, captured in an area called Orkanger also features and abundance of lamb but the sheep are of another race (Old Norwegian Spælsau) than the rest of the data.

An overview of the number of images in each sub-dataset grouped by free ranging vs. fenced, median adult sheep size and sheep race is shown in Table 9. As shown in the table, all images with extra small and extra large sheep have been excluded. This leaves images that have been captured from flight heights ranging approximately from 14 to 85 meters. The training dataset has roughly the same number of images in each sheep size category whereas the validation dataset has much fewer images in the large category than small and medium. Most of the sheep in the validation dataset are free ranging.

An overview of the number of sheep in each dataset grouped by colour and life stage is shown in Table 10. The training and validation dataset have mostly white sheep. Brown and black sheep are much less common. The T1 dataset contains almost all white sheep, whereas the T2 dataset has a much more varied colour distribution and an overweight of black sheep.

	Total	Environment		Median Adult Sheep Size					Sheep Races*			Background Only
		Free Ranging	Fenced	xS	S	M	L	xL	1	2	3	
Training	229	123	106	0	66	78	85	0	✓	✓		0
Validation	64	59	5	0	30	26	8	0	✓	✓		0
T1: Klæbu	106	0	106	0	59	40	7	0	✓			0
T2: Orkanger	116	0	116	0	16	43	35	0			✓	22
<b>Total</b>	<b>515</b>	<b>182</b>	<b>311</b>	<b>0</b>	<b>171</b>	<b>187</b>	<b>135</b>	<b>0</b>	<b>✓</b>	<b>✓</b>	<b>✓</b>	<b>22</b>

**Table 9:** The number of images in the sheep dataset after sampling. Images are grouped by distribution of sheep size and distribution of free ranging versus fenced sheep. In addition, the sheep races represented in the dataset are marked.

\* Sheep races are: **1.** Norwegian White Sheep, **2.** Norwegian Pelssau and **3.** Old Norwegian Spælsau.

	Total	Sheep Color				Sheep Life Stage	
		White	Grey	Black	Brown	Lamb	Adult
Training	2775	1555	878	257	85	0	2775
Validation	435	237	175	18	5	0	435
T1: Klæbu	1474	1422	2	28	22	409	1065
T2: Orkanger	1236	196	167	627	246	608	628
<b>Total</b>	<b>3210</b>	<b>1792</b>	<b>1053</b>	<b>275</b>	<b>90</b>	<b>1017</b>	<b>4903</b>

**Table 10:** The number of sheep in the dataset after sampling grouped by sheep colour and sheep life stage.

## 5.1.2 Image Alignment

The mean reprojection error (MRE) is used as a performance measure for the alignment process and is defined as the mean euclidean distance between some projected points and their corresponding measured points. In the case of calibration, MRE describes the mean euclidean distance between the undistorted points of the infrared chessboard images and the model chessboard grid. On the other hand, when describing the alignment process as a whole (calibration and transformation), MRE refers to the mean euclidean distance between the undistorted and transformed infrared image points and their corresponding RGB image points.

The resulting camera matrix (C) (equation 2.5) from the camera calibration process described in section 2.6 is calculated as

$$C = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 710.732 & 0 & 306.414 \\ 0 & 699.502 & 173.578 \\ 0 & 0 & 1 \end{bmatrix}$$

and distortion coefficients can be seen in Table 11. For the calibration process, the MRE is **2.124** pixels.

$K_1$	-0.30643904
$K_2$	0.1504113
$P_1$	0.01291057
$P_2$	-0.00137666
$K_3$	-0.16887709

**Table 11:** The calculated distortion coefficients

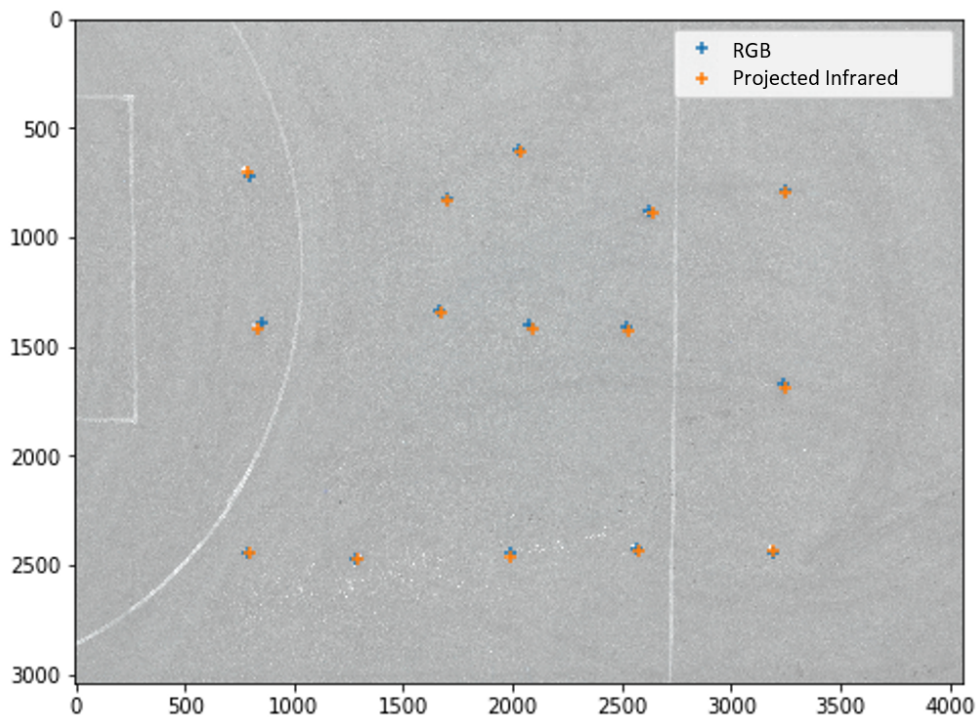
The Affine transformation matrix (T) obtained from the second part of the alignment process described in section 2.6 is

$$T = \begin{bmatrix} 5.389 & 3.970 \times 10^{-2} & 369.486 \\ -6.790 \times 10^{-2} & 5.364 & 210.352 \\ 0 & 0 & 1 \end{bmatrix}$$

This is approximately equivalent with a scaling of the image by a factor of 5.4 followed by translations of 370 and 210 pixels in the x and y directions respectively.

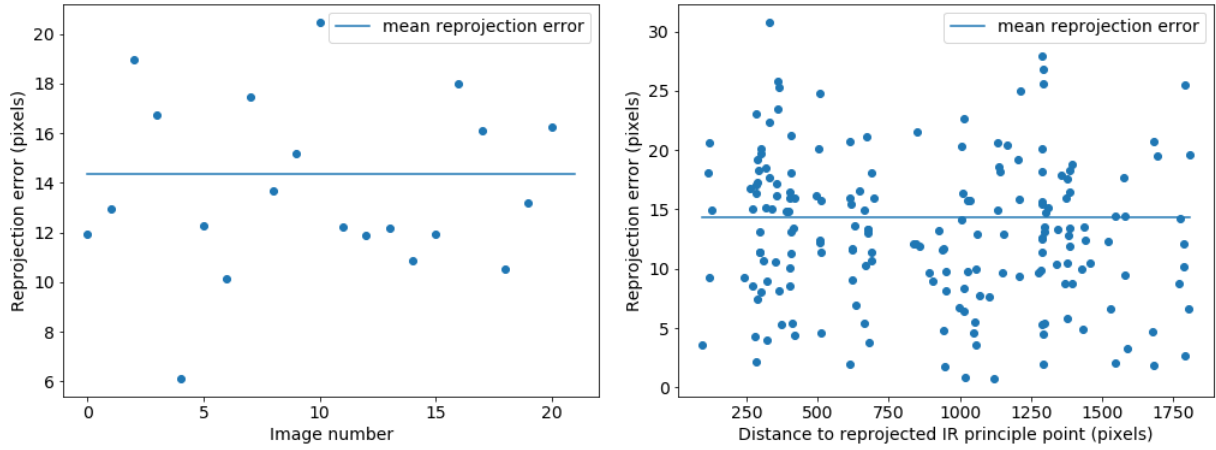
In order to evaluate the full alignment process, an independent test set of 20 images (not used in alignment calculations) are analysed. The images are also of torches as in Figure 31 but are taken on a different day and at a different location. The 20 images contain 188 feature points in total. Figure 42 shows the RGB image points and corresponding infrared points that have been projected to the RGB coordinate system from one of the test images. Another example of the alignment process is shown in Figure 44 where an aligned infrared image is masked by lines detected in the RGB image. From observing these two figures, the reprojected points or image appear to align well with corresponding features of the RGB image.

Figure 43a shows the MRE for each of the test images ordered by increasing flight height. There is no apparent relationship between the image number and MRE. Similarly, Figure 43b shows no relationship between reprojection error and the distance of feature points from the projected principle point of the infrared image coordinate system. The MRE for the full alignment process is **14.357 pixels** with a standard deviation of **6.058 pixels**.



**Figure 42:** RGB image points and reprojected infrared points

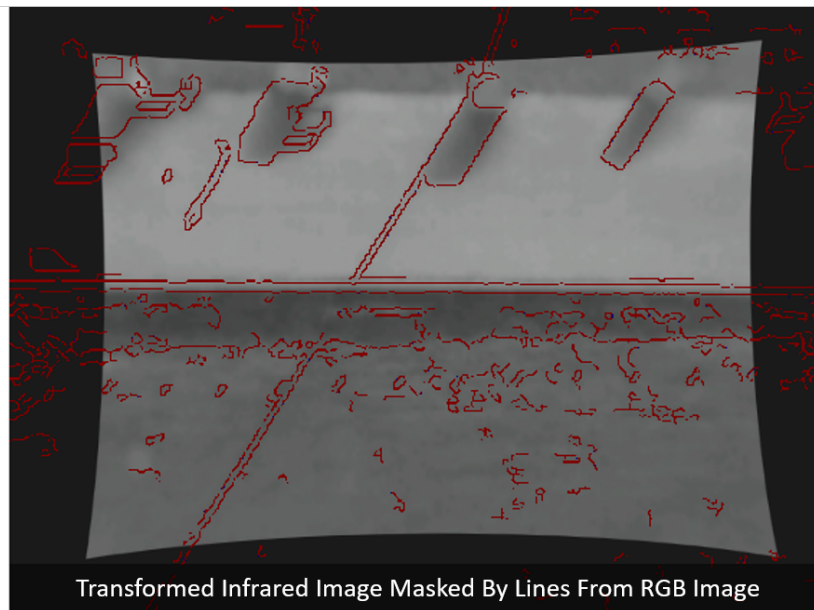
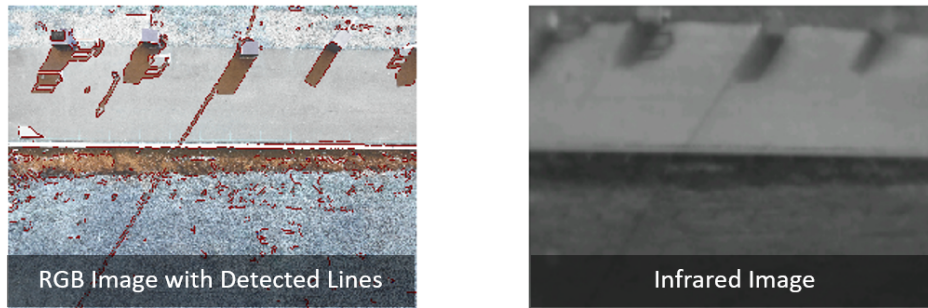
An unexpected observation can be made when the sheep in the image are moving. Figure 45 shows that boxes transformed from the RGB coordinate system to the infrared images align well when the sheep are stationary but appear in front of the sheep when the sheep are moving.



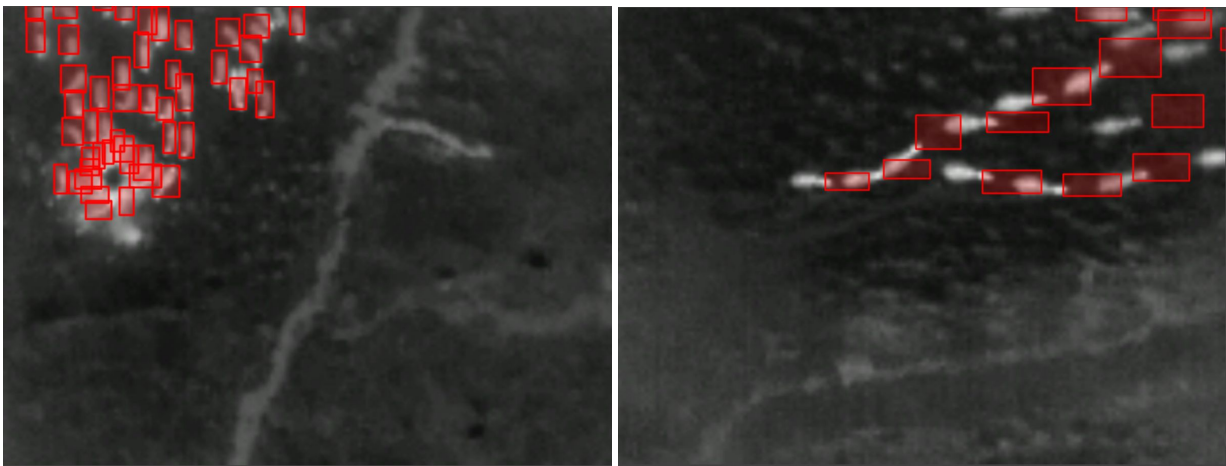
(a) mean reprojection error per test image. The test images are ordered by increasing flight height

(b) reprojection error by distance from the reprojected infrared principle point

**Figure 43:** Plots of reprojection error vs image number and distance from the image centre



**Figure 44:** Result of Image alignment shown by overlaying edge features of the RGB image onto a transformed infrared image



(a) Group of sheep standing still

(b) Group of sheep moving

**Figure 45:** Boxes from the RGB coordinate system transformed to the infrared coordinate system. Alignment appears to be good in cases when sheep are standing still but not as good in the case when the sheep are moving.

## 5.2 Experimental Results

This section presents average precision and inference time results for the various model configurations that were trained on the sheep detection task. This gives the basis on which to answer the thesis research questions. Furthermore, effects of various features of the images such as flight height and sheep colour are presented for the highest average precision model of each input datatype to give a basis for analysing the models' strengths and weaknesses.

Results of all models can be found in Table 16 of appendix A.6. Results are reported on all the sub-datasets, however the main analysis will refer to results obtained on the validation dataset. The reason for this is that the May 2020 dataset has very different properties compared to the datasets used for training and validation and is therefore not suitable to evaluate the overall model performances. The main difference between the May 2020 dataset and the training/validation datasets is the abundance of lamb and difference in sheep race. These are features of sheep that the model has not been exposed to during training so it cannot be assumed that the model will perform optimally on this dataset. Nonetheless, performance on the T1 and T2 dataset can give insight into how well the models are able to generalise to other sheep races and lamb as well as the models' ability to distinguish sheep from other unseen animals.

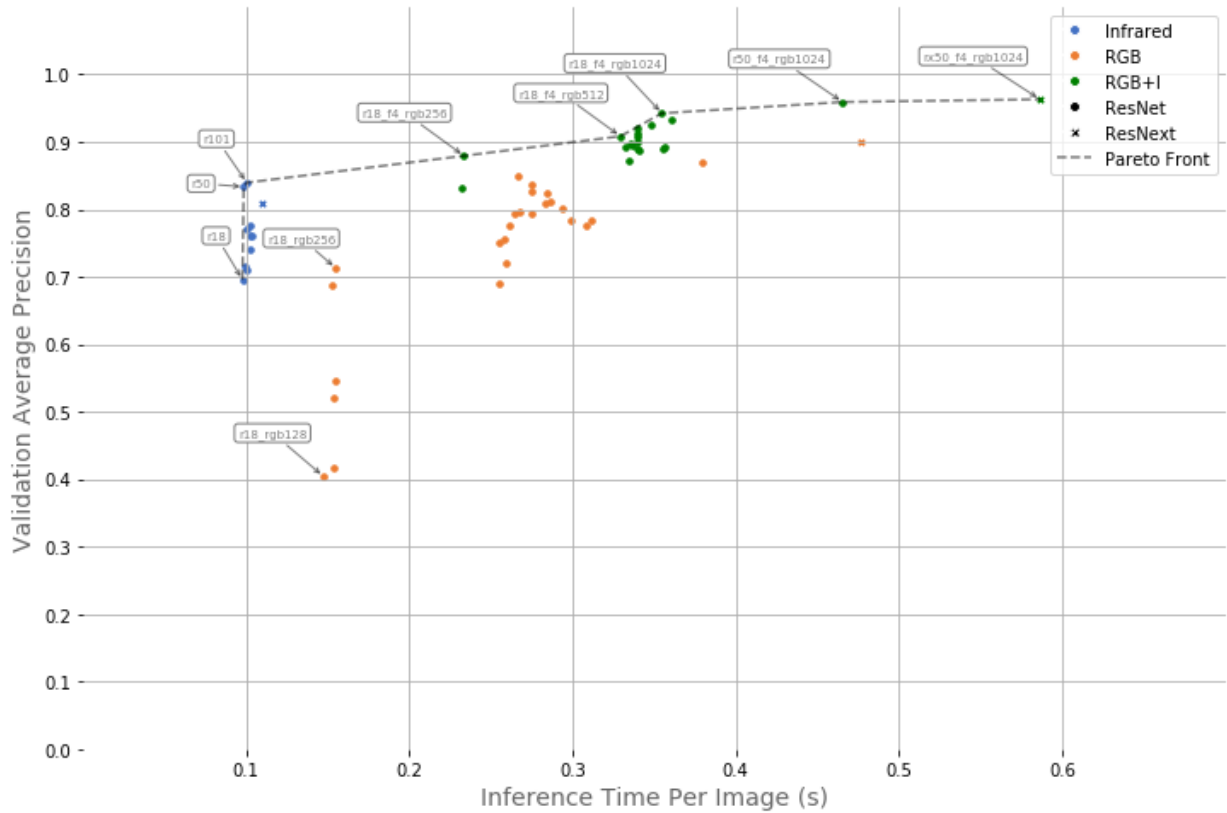
### 5.2.1 Model Configurations

Models are trained with a range of configurations in order to show the effect of input data type, network design and RGB resolution on performance metrics average precision and inference time. Validation average precision performance for all model configurations is plotted against inference time in Figure 46. Infrared only results are shown in blue, RGB only results are orange and results of the fusion (RGB+I) model are green. Some clear groups are apparent in this graph. Infrared models all have a very low inference time (around 0.1 s/image) but about 10% lower average precision than the fusion models. In comparison, the fusion models clearly have the best average precision performance (mostly above 90%) but in turn have a higher inference time. RGB models fall somewhere in between the infrared and fusion models with regards to both average precision and inference time with an overall worse trade-off between the two metrics.

There is no single optimal configuration that has both the best average precision and the best inference time performance. Instead there are a set of solutions that are optimal. The pareto front, which is marked with a dashed grey line on Figure 46 passes through the set of optimal solutions. Models that lie on the pareto front cannot be improved with regards to one of the performance metrics (AP or inference time) without sacrificing performance in the other.

Some of the points of interest (including all points on the pareto front ) are labelled on the graph in Figure 46. The models are named by model configuration settings. For example, 'r18\_f4\_rgb1024' is a model trained with a ResNet18 backbone, fuse level 4 and RGB crop resize size of 1024. Numerical results for the set of optimal points on the pareto front are also shown in Table 12. From this it can be observed that all fusion models on the pareto front have a fusion depth level of 4, which means fusion is performed after layer 'conv4\_x' in the ResNet or ResNeXt architectures (Table 2). Furthermore, it can be observed that all RGB only models are sub-optimal since they lie below the pareto line.

The highest average precision achieved by any model is 96.3%. This is achieved by the ResNeXt50 fusion model with RGB resize shape of 1024. However, the inference time cost of using this model is 0.586. On the other hand, the fastest model is a Resnet18 infrared model that has an inference time of 0.096 seconds. However, this model only achieves an average precision of 69.6%.



**Figure 46:** Validation average precision performance of all models against inference time grouped by input type. Some points of interest are labelled. Eg. 'r18\_f4\_rgb1024' is a model trained with a ResNet18 backbone, fuse level 4 and RGB crop resize shape of 1024.

Model	Fusion Depth	RGB Size	Average precision				Inference Time (s)	Grid		Sheep Recall
			Training	Validation	T1	T2		Precision	Recall	
I_r18	-	-	0.808	0.696	0.572	0.682	<b>0.098</b>	0.759	0.505	0.721
I_r50	-	-	0.893	0.833	0.481	0.585	0.098	0.846	0.726	0.871
I_r101	-	-	0.882	0.840	0.571	0.620	0.100	0.844	0.689	0.856
RGB+I_r18	4	256	0.983	0.879	0.593	0.912	0.233	0.834	0.807	0.925
RGB+I_r18	4	512	0.985	0.908	0.835	0.924	0.329	0.839	0.835	0.948
RGB+I_r18	4	1024	0.991	0.942	0.891	0.939	0.355	0.901	0.858	0.960
RGB+I_r50	4	1024	0.990	0.959	0.894	0.940	0.465	0.959	0.873	0.968
RGB+I_rx50	4	1024	<b>0.993</b>	<b>0.963</b>	<b>0.900</b>	<b>0.945</b>	0.586	<b>0.979</b>	<b>0.901</b>	<b>0.975</b>

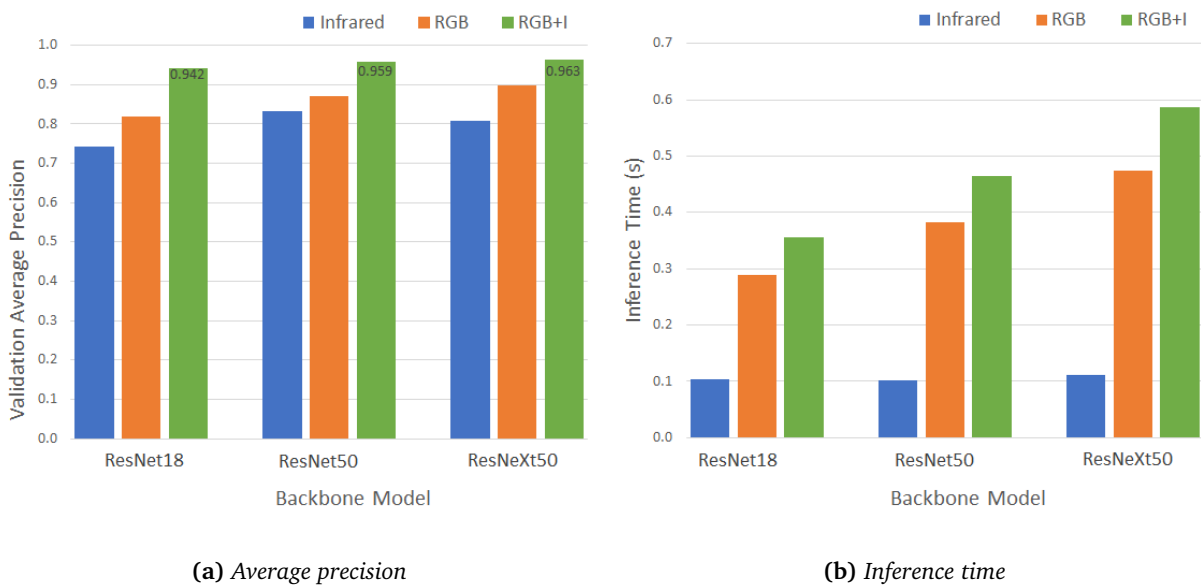
**Table 12:** Average precision, precision, recall and inference time results for models on the pareto front. The best values are highlighted in bold. Grid precision, grid recall and sheep recall are for the validation dataset at a confidence threshold of 0.5



In order to analyse the effect of input type and model design on performance, models with fixed RGB and infrared crop sizes of 1024 and 64 respectively and fixed fusion depth of 4 where applicable are shown in Table 13. Figures 47a and 47b show bar plots of the effect of input type and network design on average precision and inference time respectively. From observing Table 13, it is apparent that the deepest high cardinality backbone model ResNeXt50 with RGB+I input gives the best average precision result. However, from observing Figure 44, input type appears to have a much larger impact on average precision than network design, with the fusion model outperforming both RGB and infrared models for every model backbone. Similarly, for inference time input type appears to have a larger impact than backbone model with the infrared model being more than three times as fast as the RGB and fusion models in most cases. The infrared model also appears unaffected by increased network depth and cardinality, whilst both RGB and RGB+I models gain about 0.1s in processing time when increasing model depth from 18 to 50 or increasing cardinality from 1 (ResNet50) to 32 (ResNeXt50).

Model Backbone	Input	Average Precision				Inference
		Training	Validation	T1	T2	Time (s)
ResNet18	Infrared	0.880	0.741	0.705	0.435	0.100
	RGB	0.925	0.817	0.843	0.708	0.285
	RGB+I	0.991	0.942	0.939	0.804	0.355
ResNet50	Infrared	0.893	0.833	0.585	0.365	<b>0.098</b>
	RGB	0.936	0.871	0.852	0.801	0.380
	RGB+I	0.990	0.959	0.940	0.810	0.465
ResNeXt50	Infrared	0.909	0.808	0.770	0.440	0.109
	RGB	0.970	0.899	0.918	0.818	0.477
	RGB+I	<b>0.993</b>	<b>0.963</b>	<b>0.945</b>	<b>0.823</b>	0.586

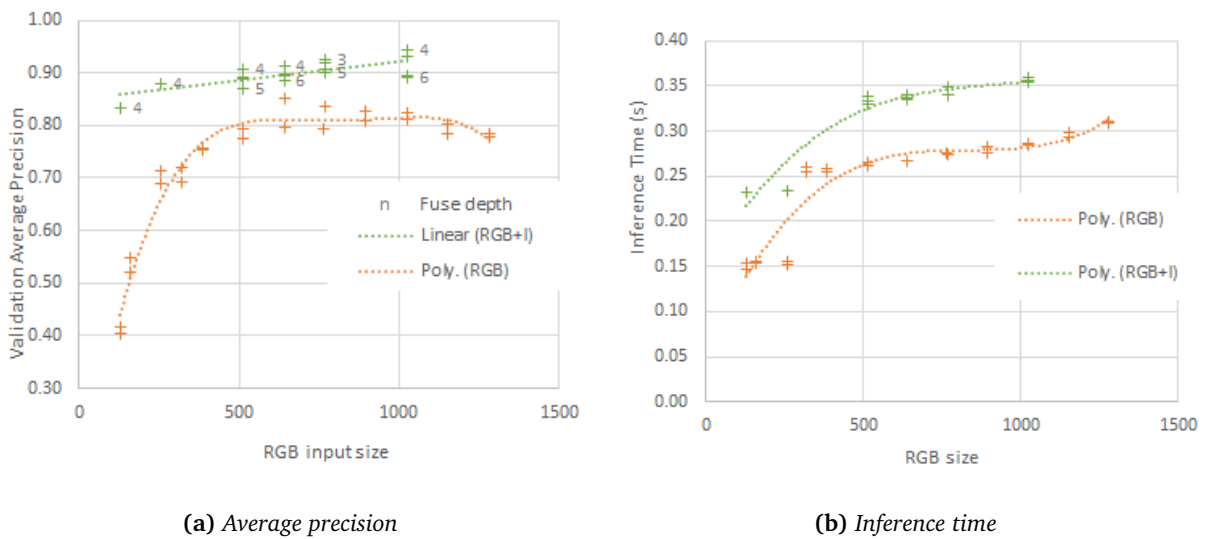
**Table 13:** Result of models grouped by model backbone and input type. All models shown in this comparison are trained on RGB crop resize shape of 1024, infrared crop resize shape of 64 and fused at depth 4 (for RGB+I). An average is taken in the cases where more than one model with the same configuration exists. The best values are highlighted in bold.



**Figure 47:** Validation average precision and inference time grouped by model backbone and input type. Numerical values for this graph can be found in Table 13.

The effect of RGB resolution on average precision and inference time is presented in Figure 48a and 48b respectively. All experiments in varying RGB resolution are done with a Resnet18 backbone. RGB input size is the size that the raw  $1200 \times 1200$ -pixel RGB image crop is resized to prior to being processed by the model. Results of the RGB only model is orange and the fusion model (RGB+I) is green. In addition, the best and worst fuse depth for each RGB size is marked for the RGB+I plot.

For the RGB model there is a clear drop in average precision performance when the RGB resize size is lower than 512 (43% resolution), however this drop is not apparent for the RGB+I model. The RGB+I model has a more linear trend with a higher RGB resolution yielding a higher average precision performance. Fusion depth 4 has the highest average precision for 75% of the RGB resolutions and fusion depth 5 and 6 each have the worst average precision results in half the cases. With regards to inference time (Figure 48b), there is a general trend of increased inference time with increased RGB resolution with a distinct drop in processing time when RGB sizes are lower than 320 pixels for both the RGB and RGB+I models. Very similar inference time results are achieved by the different fusion depths at each resolution model for the RGB+I model.



**Figure 48:** The effect of RGB resolution on validation average precision and inference time performance for the RGB and fusion (RGB+I) models with backbone model ResNet18. RGB input size is the size that the raw  $1200 \times 1200$ -pixel RGB image crop is resized to prior to being processed by the model.

## 5.2.2 Best Model Performance

This section takes a closer look at the performance of the models that achieve the highest average precision for each input data type. These are the ResNeXt50 models with RGB resize size 1024. The effects of median sheep size, sheep colour and sheep life stage are presented. A small selection of sample predictions for the three input types is shown. In addition, some sample predictions by the RGB+I ResNeXt50 model on the validation, T1 and T2 datasets can be seen in appendices A.3, A.4 and A.5

### Grid Precision and Recall

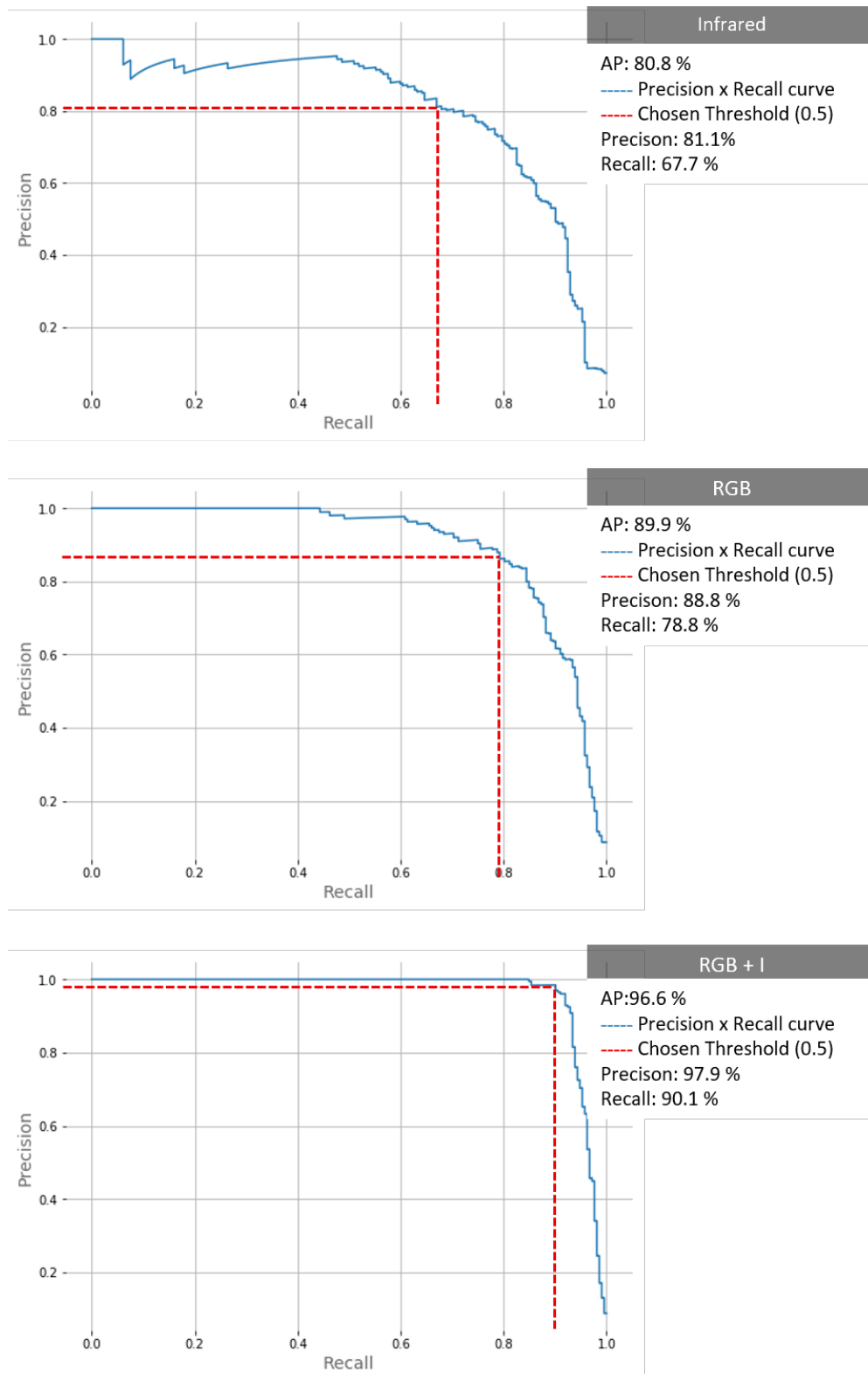
Precision and recall for the models on the validation dataset is plotted in precision  $\times$  recall curves in Figure 49. These values can also be seen in Table 14. The red line on the plots shows the precision and recall values obtained when a confidence threshold of 0.5 is chosen. 0.5 appears to give a good trade-off between precision and recall since the line reaches the curve close to its sharpest. The RGB+I model achieves a recall of 90.1% with a precision of 97.9%. This means that the model is able to detect 90.1% of all the grid cells that contain sheep with 97.9% of the detections being correct detections. In comparison, the RGB and infrared only models perform much worse, with both achieving more than 9% worse precision and recall

scores than the combined model. The inferior performance of the infrared and RGB models compared to the RGB+I model is also clear from the more rounded shape of the precision  $\times$  recall curve.

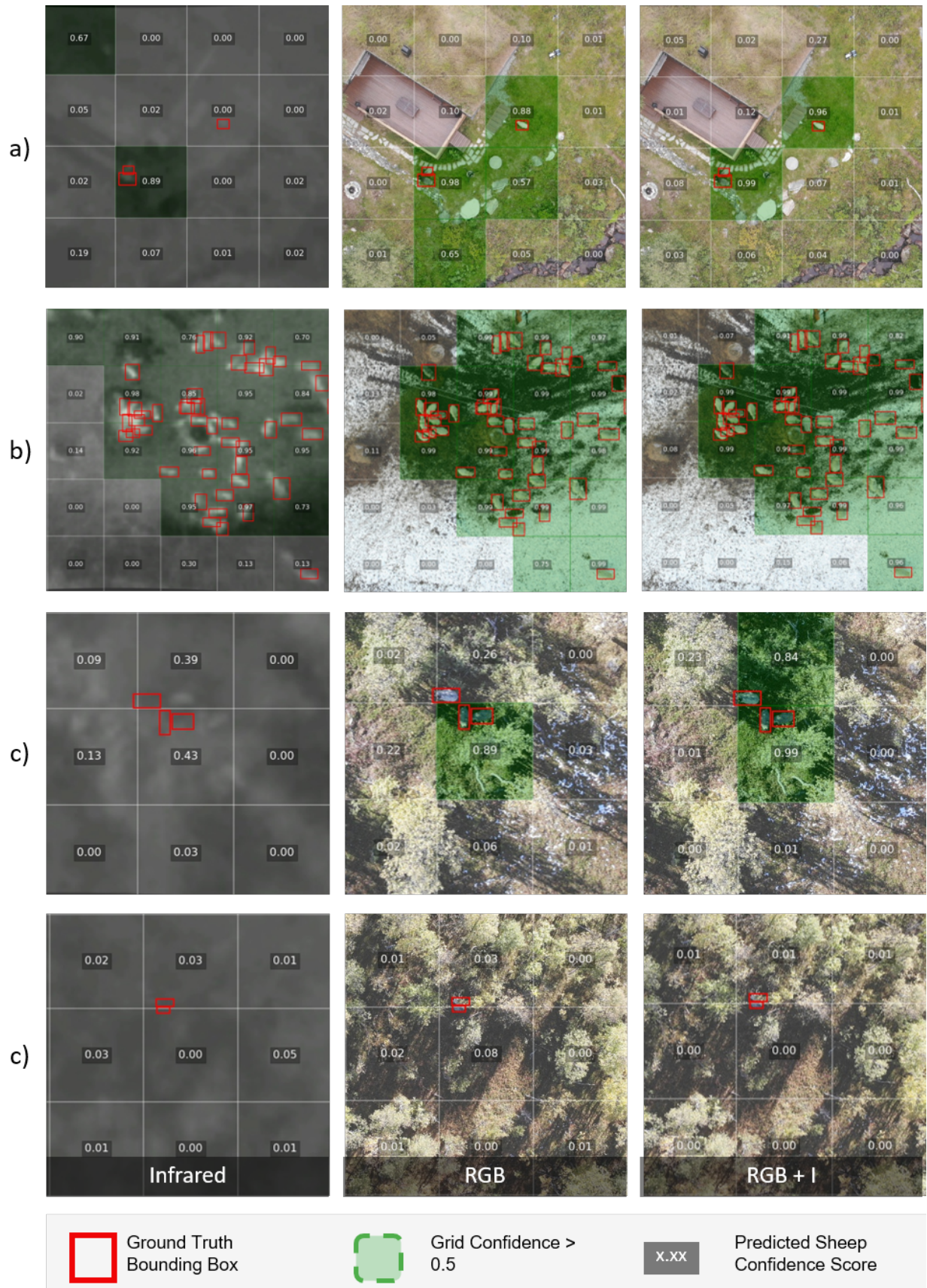
		AP by Median Sheep Size					
Input Type	Dataset	All	S	M	L	Precision	Recall
Infrared	Training	0.909	0.821	0.932	0.930	0.907	0.791
	Validation	0.808	0.648	0.875	0.858	0.811	0.670
	T1	0.770	0.767	0.817	0.915	0.793	0.661
	T2	0.440	0.380	0.569	0.354	0.440	0.517
RGB	Training	0.970	0.950	0.950	0.972	0.930	0.921
	Validation	0.899	0.846	0.927	0.872	0.888	0.788
	T1	0.918	0.905	0.949	0.922	0.908	0.838
	T2	0.818	0.766	0.866	0.865	0.769	0.759
RGB + I	Training	0.993	0.985	0.994	0.997	0.973	0.960
	Validation	0.963	0.918	0.984	0.968	0.979	0.901
	T1	0.945	0.942	0.964	0.995	0.942	0.885
	T2	0.823	0.761	0.883	0.860	0.834	0.726

**Table 14:** Average precision, precision and recall for ResNeXt50 models. Average precision is grouped by median adult sheep size per image as defined in Figure 20. Grid precision and recall is calculated with using a confidence threshold of 0.5.

From looking at the sample detections shown in Figure 50 it appears that the RGB and infrared models have different strengths and weaknesses that complement each other in such a way the combined model result is able to benefit from both information sources. For example image set a) in Figure 50: the infrared model correctly classifies one grid as having sheep but misses the other. The infrared model also mistakenly classifies the top left grid as having sheep. On the other hand, the RGB model finds both grids containing sheep but also has two false positive classifications. Both the infrared and the RGB models predict false positives but they predict them in different areas. The RGB+I model is able to correctly classify the grids containing sheep but without making the same mistakes as the single input models. Image sets b) and c) in Figure 50 shows more examples of this phenomenon. Image set d) in Figure 50 shows an example of some sheep that none of the models were able to detect.



**Figure 49:** Precision×Recall curve for the ResNeXt50 I, RGB and RGB+I models. The red shows which precision and recall values that are obtained when selecting a threshold of 0.5.

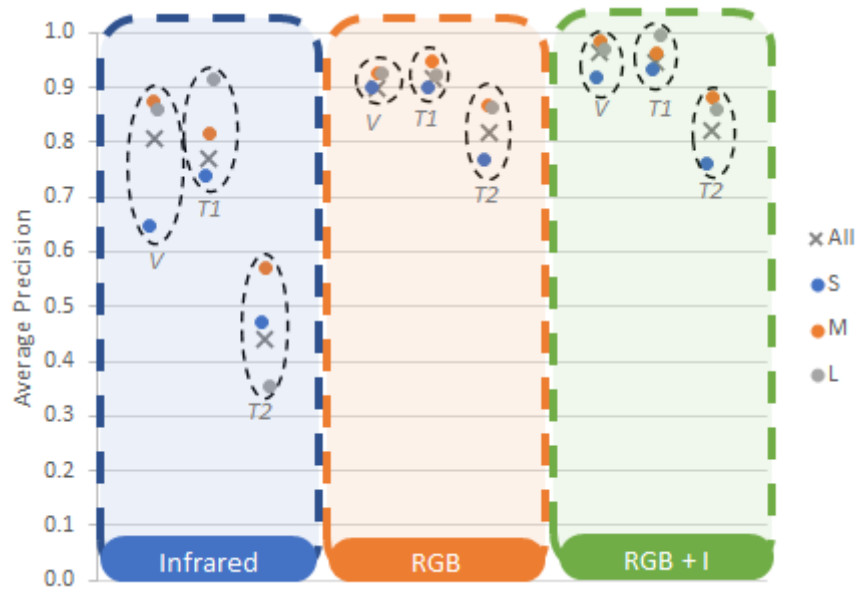


re

Figure 50: Examples of predictions made on the validation dataset by the ResNeXt50 I, RGB and RGB+I models.

## Median Adult Sheep Size

Median adult sheep size as defined in Figure 20 is an important factor for average precision performance. Figure 51 shows that all models with the exception of the infrared model on the T2 dataset performs worst on images with the sheep size category S. Table 14 shows the numerical values for this plot. Median adult sheep size is a proxy measure for UAV flight height and by estimations explained in section 3.3 another phrasing of this observation is that images taken from a flight height of more than approximately 43 meters have a worse average precision result than images taken from flight heights ranging from approximately 14 to 43 meters. Looking at validation results, which are used to benchmark the model performance, M sheep size images do slightly better than the L images. M images also perform better than the L images in several of the T1 and T2 datasets. As a result, there is no evidence to suggest that improved results are gained from flying lower than 28 meters when photographing the sheep. The difference in performance by sheep size is greater for the infrared model than the RGB and RGB+I model.



**Figure 51:** Average Precision grouped by dataset and median adult sheep size for RGB + I model

## Sheep Recall

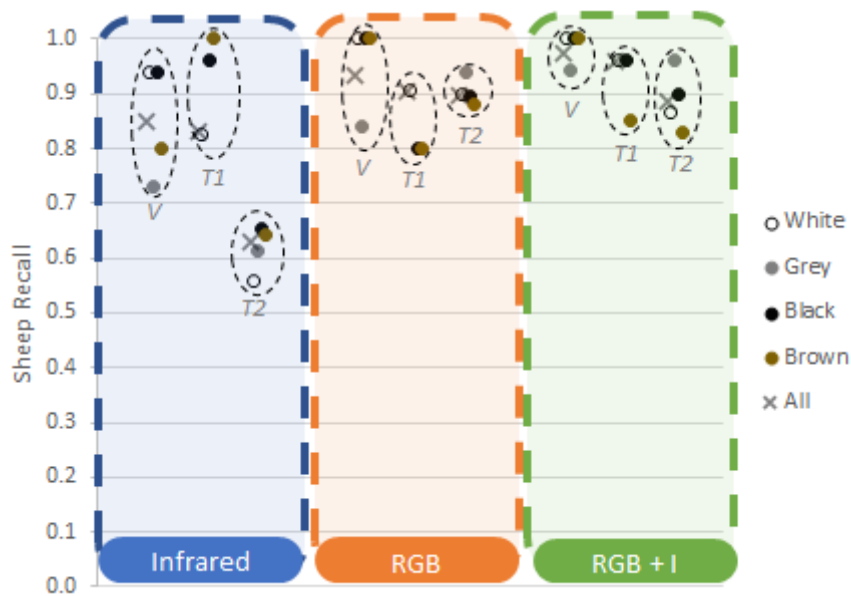
Sheep recall is the proportion of all the sheep that are detected by the model at a confidence threshold of 0.5. A sheep is considered detected (true positive) if any part of its bounding box overlaps with a grid cell with sheep confidence score greater than 0.5. On the other hand, if a ground truth bounding box does not overlap with any grid cell with a sheep confidence score greater than 0.5 then this sheep is not found (false negative). Using this metric, 97.5% of all the sheep in the validation dataset were detected by the RGB+I model. Table 15 shows sheep recall values obtained by the ResNeXt50 models grouped by sheep colour and sheep life stage.

## Sheep Recall by Sheep Colour

Figure 52 shows sheep recall for the various datasets grouped by colour. Infrared images do not have colour so as expected, there is also no apparent relationship between sheep colour and sheep recall. A more surprising result is that there is also no clear relationship between sheep colour and average precision performance for the RGB model. For the validation data set, perfect recall scores are obtained for white, black and brown sheep. It should however be noted that the validation dataset only 18 black and 5 brown sheep in the dataset compared to 237 white sheep.

Input Type	Dataset	Sheep Colour					Life Stage	
		All	White	Grey	Black	Brown	Adult	Lamb
Infrared	Training	0.901	0.869	0.947	0.926	0.949	0.901	-
	Validation	0.848	0.938	0.731	0.938	0.800	0.848	-
	T1	0.830	0.824	-	0.960	1.000	0.788	0.934
	T2	0.631	0.557	0.612	0.654	0.642	0.686	0.573
RGB	Training	0.986	0.985	0.984	0.986	1.000	0.986	-
	Validation	0.933	1.000	0.842	1.000	1.000	0.933	-
	T1	0.903	0.907	-	0.800	0.800	0.914	0.876
	T2	0.898	0.899	0.938	0.896	0.881	0.935	0.860
RGB + I	Training	0.991	0.988	0.996	0.986	1.000	0.991	-
	Validation	0.975	1.000	0.942	1.000	1.000	0.975	-
	T1	0.961	0.963	-	0.960	0.850	0.958	0.968
	T2	0.887	0.867	0.961	0.898	0.830	0.948	0.821

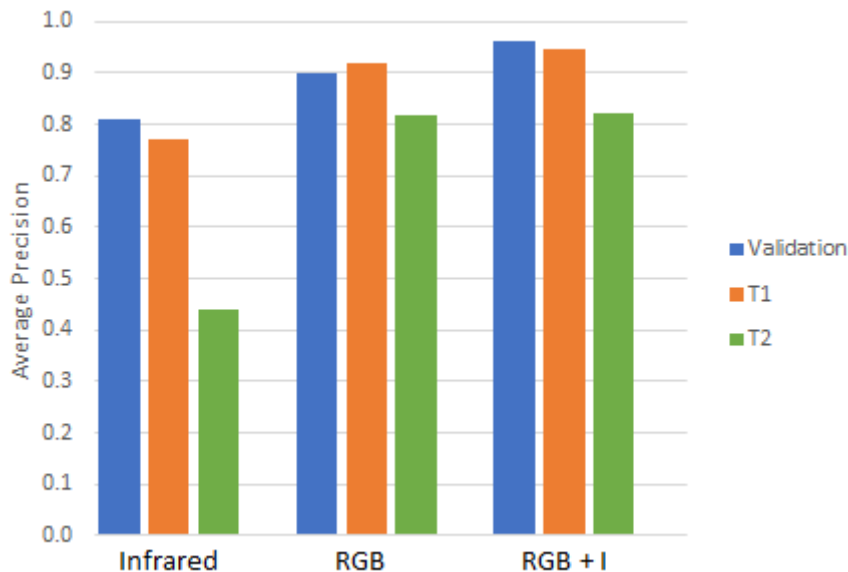
**Table 15:** Sheep recall grouped by input type, sheep colour and sheep life stage.



**Figure 52:** Sheep Recall by colour.

### 5.2.3 Case Studies

Model performances on datasets T1 and T2 are analysed in this section to provide insight into the models' generalisation ability to lamb and an unseen sheep race. Both T1 and T2 datasets feature lamb however dataset T2 also features Old Norwegian Spælsau, which is a particularly hairy and colourful race that the networks have not been exposed to during training. In addition, these datasets also feature some other animals. The models' ability to distinguish sheep from other animals will also be presented here. Comparisons of performance between the validation, T1 and T2 datasets can be seen in Figure 53. The models generally perform worse on the T2 dataset than the T1 and validation datasets. The infrared model performs especially poorly (AP 44%) on the T2 dataset. In comparison, the T1 dataset achieves similar average precision results to the validation dataset.



**Figure 53:** Average precision performance of the best Infrared, RGB and Fusion models on the Validation, T1 and T2 datasets

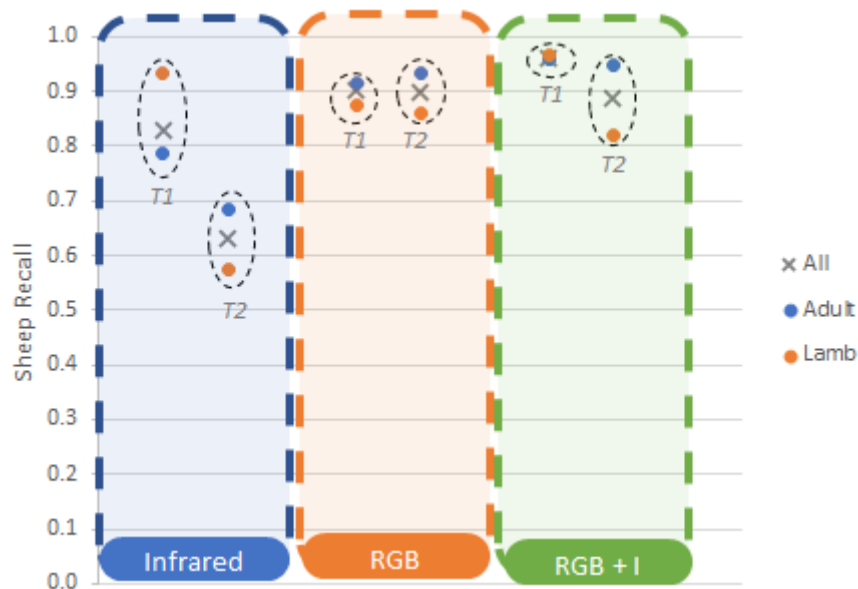
#### Generalisability to Lamb

Figure 54 shows sheep recall values achieved for lamb compared to adult sheep in the T1 and T2 datasets. Numerical values for these plots can be found in Table 15. For the T2 dataset, lamb recall is consistently lower than adult sheep recall. However, for the T1 dataset, lamb recall is actually higher than adult recall by approximately 10% for the infrared model and about equal for the RGB and fusion models.

#### Generalisability to Unseen Race

The T2 dataset contains images of Old Norwegian Spælsau. The very hair and colourful characteristics of the Old Norwegian Spælsau distinguish them from the images of Norwegian Pelsau and Norwegian White Sheep in the other datasets. As a result, comparing performance on T2 with performance on the validation and T1 datasets can give insight into the models' generalisation ability to a new sheep race. However, it should also be noted that the abundance of lamb also present in T2 is a confounding factor on the difference in performance between the T2 and the validation dataset. Difference in average precision performance between the validation, T1 and T2 datasets is shown in Figure 53. From observing this figure, it is clear that models perform worse on the T2 dataset for all data types but the difference is especially large for the infrared model. The infrared model achieves a 44.0% average precision on T2 compared to the RGB model, which achieves almost twice as good results (81.8%) on T2. There is less than 1% average precision gain achieved by using the RGB+I model on T2 compared to RGB only.





**Figure 54:** *The difference in sheep recall for lamb vs. adult sheep*

### Other Animals

A qualitative presentation of the RGB+I ResNeXt50 model's performance with regards to distinguishing sheep from other animals is shown in Figure 55. Animals in this Figure all come from the T1 or T2 dataset. The model's distinguishing ability is classified for each animal as one of three categories: 'correct', 'mostly correct' or 'wrong'. The 'correct' classification means that the model correctly did not classify the animal as a sheep in all cases, the 'wrong' classification is given if the grid containing the animal was incorrectly classified as sheep in the majority of cases and finally, the 'mostly correct' classification is given if the model got the correct classification for these cells in the majority of cases. Results in Figure 55 show that the model has no problem distinguishing small animals (chickens, peacocks and ducks) from sheep. Hairy pigs and humans are mostly correctly classified whereas horses were consistently wrongly classified as sheep.



Figure 55: How well the model distinguished other animals in the dataset from sheep

## 6 Discussion

The goal of this thesis is to develop a deep learning model that automatically detects and locates sheep in dual RGB and infrared UAV images and evaluate the extent to which the model meets detection performance and processing speed requirements of a real-world application. In particular, the research questions explore the effects of varying input data type (RQ1), model design (RQ2) and input resolution (RQ3) on model average precision and inference time performance. This chapter discusses the results obtained from following data processing and experimental methods outlined in chapter 4 in light of these research questions, relevant theory and related work. Moreover, the practical implications and limitations of the work as well as recommendations for future work is discussed.

### 6.1 Data Preprocessing

The results of data sampling and alignment are discussed in this section. Moreover, the implications of challenges met, and the decisions made to overcome them are discussed.

#### 6.1.1 Data Sampling

Data sampling by criteria explained in section 4.1.1 resulted in a dataset consisting of 515 images. However, only 293 of these images were available at the time when training of the network was being done. These 293 images were captured in 20 different sessions where images from the same session are not independent since they show the same group of sheep interacting in the same environment. Unfortunately, due to this limited amount of independent data, a representative test dataset for unbiased evaluation of the model performance could not be made without compromising the quality of the training and validation datasets. It was intended for data collected in May 2020 to be used for a test dataset, however the distribution of this new data was too different from the training and validation data that it would not be reasonable to use it to benchmark the trained models' performances. The lack of a test dataset is a limitation to the validity of the results and future work should aim to collect more data so that an independent sub-dataset for testing can be used to reliably evaluate model performance.

When dividing the data into training and validation sub-datasets, an effort was made to balance the datasets in terms of median adult sheep size and sheep colours. However, some categories remain underrepresented in the validation dataset. These are images with large median adult sheep size (flight height lower than 28 meters) and images containing black and brown sheep.

The decision to exclude MSX format infrared images is justified for the fusion model since useful features of the RGB image will anyways be considered by the RGB branch of the fusion network. On the other hand, future research could consider MSX format images for the single input model as this could give a better average precision vs. time trade-off than the single input regular infrared model. This is because a single input MSX model will be just as fast as the infrared model since the resolution of the two infrared formats are the same but average precision performance will potentially be higher since there is more information encoded in the MSX image. Unfortunately, the current dataset has majority regular infrared images and these cannot be transformed to MSX format using FLIR software tools, [78], since the M2ED UAV saves images in regular JPEG format instead of radiometric JPEG format.

#### 6.1.2 Image alignment

Considering that the infrared camera has a sensor resolution of  $160 \times 120$  and the resolution of the visual images is  $4056 \times 3040$ , a perfect mapping between the two images cannot be expected. In fact, the expected

reprojection error due up-scaling of the image alone can be up to 12.66 pixels (equation 6.1). As a result, a total MRE of  $14.357 \pm 6.058$  is acceptable.

$$E(MRE)_s \approx \frac{1}{2} \times \frac{4056}{160} = 12.66 \quad (6.1)$$

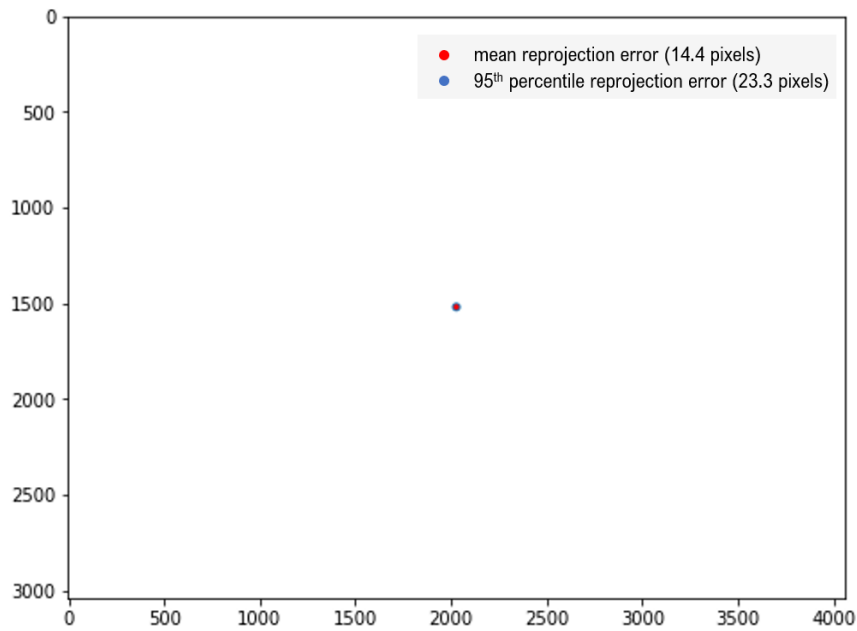
The test images are ordered by increasing flight height. Figure 43a shows no apparent relation between MRE and image number. Therefore, there is no reason to assume a change in MRE with varying flight height. In addition, it appears that the MRE distribution is the same along the edges of the images as in the centre (figure 43b). This suggests that the undistortion process was successful.

In order to put the reprojection error in perspective, the size of the MRE relative to the size of the visual image is shown in Figure 56. If the true position of the point is in the centre of the circles, the reprojected point will on average fall somewhere inside the red circle and in 95% of the cases within the blue circle. Considering the low localisation quality requirement of the sheep detector, this an acceptable error.

The gross misalignment observed when the sheep are moving fast in their environment indicates that there is a time delay between capturing the infrared and the RGB image. This effect can be reduced by not flying the UAV too low as this will scare the sheep and cause them to run. If the misalignment is small (since sheep are moving slowly in their environment), the misalignment could possibly be a useful feature as it would indicate the presence of a living animal in contrast to its stationary environment.

The necessity of undistorting and aligning the infrared image before feeding it to the CNN can be questioned since it could be argued that the deep neural network involved with sheep detection would be able to learn this relationship. Moreover, the various down-sampling and up-sampling layers of the network will reduce the effect of the image distortion. On the other hand, this relationship could only be learned by the CNN if the full image was fed to the network during training, which is impractical due to memory considerations and would reduce the great variation benefit gained from taking random rotated and flipped crops of the image during training.

If the alignment process was to be repeated, it could be a good idea to look into automating the selection of matching feature points as this would save a lot of time. Perhaps using a circular calibration pattern and finding a way to get a sharper and more distinct temperature difference would improve the alignment process. Moreover, if the quality of images in the calibration process was better, then these could also be used for the calculating the final affine transformation instead of using the separate images of torches.



**Figure 56:** The size of the reprojection error relative to the size of the image

## 6.2 Model Configuration Experiments

A series of models with various configurations with regards to input type, model design and image resolution were tested for average precision and inference time performance. The goal of doing this was to answer the research questions and find the model configurations that give the best average precision vs. inference time trade-off.

### The Optimal Solutions

The configurations that fall on the pareto front of the average precision vs. time plot (Figure 46) are the set of optimal solutions that give the best trade-off. As explained in section 5.2.1, this is because no improvement can be made to one of the performance metrics without detrimental consequences to the other. From observing which configurations fall on the pareto front, it is clear that fusion of RGB and infrared data in a single model yields better average precision performance scores than using the two inputs separately. This is consistent with results of previous research into multi-channel image fusion in deep learning models, [6, 7], and is expected since this fusion model has access to more information. From observing Figure 50, it appears that the RGB and infrared images often complement each other by endorsing each other's correct detections and opposing each other's mistakes. The most clear example of this is image set a) of Figure 49, where the infrared model has a correct prediction but also mistakenly predicts the top left corner as containing sheep (likely confusing the warm chimney with sheep). At the same time, the RGB model has correctly found the grid cells that contain sheep but is relatively confident that some cells with misleading rocks in fact contains sheep. The RGB+I model, which has both information sources available is thus able to correctly find the cells that contain sheep without making the same false positive predictions as the single input models.

When it comes to fusion depth, fusion depth 4 performs better than other fusion depths. Fusion depth 4 means that fusion of RGB and infrared feature maps is performed after layer 'conv\_4' in the ResNet or ResNeXt architectures (Table 2). This is consistent with findings by Hassan in his experiments with fusion at various depths in the ResNet model for segmentation purposes, [50].

Infrared only models also appear in the set of optimal solutions because of the fast inference time that cannot be matched by other models. This is expected due to their low resolution, which means much fewer operations are required to process the image. However, the maximum average precision achieved by any infrared only model is 84%, which is more than 10% less than the best fusion model. As mentioned in the previous section, future research could consider a single input MSX format infrared model as this would likely match the low inference time of the regular infrared model but achieve a higher average precision score due to increased information available from the embedded edge features in the image.

For the validation dataset, all RGB models are sub-optimal since their results lie well below the pareto line. This suggests that it is preferable to reduce inference time by other means (e.g. by reducing RGB resolution or model complexity) than excluding infrared features from the computations.

### Independent Variable Effects on Average Precision

Input type is clearly important for average precision performance. From observing Figure 47a, it can be observed that despite slight improved average precision performance from more complex models, input type has a much larger effect. An explanation for this is that a model is only as good as the available data. While a deeper and higher cardinality model has the capability to discover more complex and abstract features of the input, this is dependent on those features existing in the first place.

A similar argument can be made for the pattern seen when observing the impact of RGB resolution on average precision performance in Figure 48a. When the input resolution is reduced, the available information to the network is also reduced. There is a clear drop in average precision performance for the RGB model when the input resolution is lower than 43%. It is possible that at this level of down sampling some defining features of 'sheepness' (e.g. details like texture, ears and tails) is lost, which makes it difficult for the model to distinguish sheep from other objects. In comparison, this pronounced drop in performance at lower RGB resolutions is not present for the RGB+I model, which is likely due to the fact that these

models can also lean on information from the infrared image to support their predictions.

### Independent Variable Effects on Inference Time

Input type is also an important factor for inference time (Figure 47b), with the infrared model being around 3 times faster than the RGB model. The relatively large difference in inference time between the infrared and RGB model is most likely because of difference in resolution since the exact same CNN architecture and data loading process is used for both RGB and infrared models. The plot of RGB resolution against inference time (Figure 48b) confirms the large impact of image resolution on inference time, however the distinct drop in inference time for RGB sizes lower than 320 pixels is interesting. It is difficult to know what is causing this sudden drop in processing time since there are several parallel processes involved that contribute to the inference time (Figure 41). However, the large number of workers needed for loading higher resolution images indicate that some part of the data loading process is demanding a lot of computational resources. In fact, for resolutions larger than 320 pixels, no further gain is obtained from further increasing the number of workers, which may indicate that some limitation in hardware capacity has been reached and is bottlenecking the processing time. Further work to identify which parts of the inference process are contributing the most to inference time would give useful insights to help understand this pattern and find ways to reduce inference time further.

In terms of the effect of model backbone on inference time, the RGB and RGB+I models gain processing time with increasingly more complex models, while the infrared model appears unaffected by the change in model design. It is expected that a deeper model will have a higher inference time since more operations must be performed to process the image. An explanation for why this effect is not seen for the infrared model could be that processes on the machine's CPU involved with loading and preparing data are taking longer than the CNN model computations. Alternatively, the difference in processing time on the low resolution infrared image between the various model designs could simply be so small that it is not apparent from the tests.

### A Closer Look at Model Performance

Results of the ResNeXt50 models for the three input types were presented in greater detail to give an idea of which properties of the image affected model performance. Knowing the strengths and weaknesses of these models can help give ideas on measures that can be taken during data capturing to assure better results and allow the end-user of the model to be aware of the model's limitations.

The impact of median adult sheep size in images is shown as a proxy measure for UAV flight height. Results in Figure 14 show that images with a median adult sheep size category S had worse average precision performance than images with categories M or L. This is an expected result because a higher sheep resolution entails the presence of more distinct sheep features. The fact that this difference is much larger for the infrared model than the RGB model suggests that the low resolution of the infrared camera is the greatest limitation to how high the UAV should fly when capturing images. The practical implication of this result is that when data is collected using this particular UAV, the pilot should aim to keep a flight height under 43m. The results do not indicate that L images achieve any better result than M images, which means that there is no evidence to suggest that performance gains can be made from flying lower than 28 meters. However, the validation dataset only has 8 images in the L size category, which is not enough to draw a conclusion from this observation. Future research could consider analysing UAV flight height more systematically to find an acceptable trade-off between flight height and detection performance. Being able to capture data from greater heights is beneficial because it allows for a larger area to be covered in fewer images and consequently an area of interest may be surveyed in a shorter flight time. Moreover, it should be considered to incorporate flight height information to the model as suggested by Muribø, [22], as this will give the model clues to the expected size of the sheep in the images and can help to exclude some false positive predictions.

Results in relation to sheep colour were somewhat unexpected. Previous research on automatic sheep detection in RGB UAV images, [21, 23], found that white sheep were easier to find than other colours. As a result, a similar trend is expected for the RGB models. This was however not the case. The correlation is not

apparent in the results, but that does not mean that a relationship between sheep colour and performance does not exist. For one, these results must be seen in perspective with regards to the prevalence of each sheep colour in the dataset. For instance, the validation dataset only contains 18 black and 5 brown sheep, which is not enough to draw conclusions about the model performance on all sheep of that colour. Furthermore, the effect of sheep colour on average precision is convoluted by the fact that the model does not predict individual sheep but rather the probability of sheep presence within a certain area. As a result, multiple sheep of various colours can be detected by a single positive prediction by the model.

Datasets T1 and T2 are included in the results as a case study to assess how the model generalises to lamb, an unseen sheep race and other animals. The model achieved comparable scores on the T1 dataset compared to the validation dataset, however this does not necessarily mean that the model generalises well to lamb since the dataset is also full of other confounding factors that could contribute to its high performance. The lamb in the T1 dataset mainly stayed close to the adult sheep so it could be the case that the model is mainly detecting the adult sheep and the lamb just happen to be located in the same grid. Moreover, all sheep in the T1 dataset are grazing in an open fenced field, which means there are fewer other objects to confuse the sheep with and sheep occlusion does not occur to any large extent.

The models perform much worse on the T2 dataset than other datasets. The grazing environment in the T2 dataset is more similar to the free grazing environment with many rocks and other objects present. This could be one factor that influences the poorer result, however the model has been exposed to similar types of environments during training so the notably poorer performance is most likely attributed to properties of the unseen sheep race and/or the presence of lamb. In contrast to T1, lamb in the T2 dataset did not always walk so close to an adult sheep, which could contribute to the poorer performance. The disparity in detection results between lamb and adult sheep in the T2 dataset shown in Figure 54 supports this claim. The properties of the sheep race in T2 could also explain the inferior performance. The most notable difference in performance between T2 and the other datasets is by the infrared model. This could be explained by the presence of lamb, which are smaller making them less likely to get detected by the low-resolution infrared camera. Moreover, the very hairy nature of the Old Norwegian Spælsau in the image could be insulating the infrared radiation and reducing its strength.

The model's predictions on certain other animals in the T1 and T2 datasets show that distinguishing sheep from other animals is a challenge. Although it is not likely to encounter these particular animals in the sheep's free grazing terrain, the model's trouble with distinguishing sheep from horses and hairy pigs does give reason to suspect that it would make similar mistakes if encountering moose, deer, dogs or other animals.

### 6.3 Practical implications and Limitations

The highest average precision model trained on the sheep dataset was able to detect 97.5% of all the sheep in the validation dataset with 97.9% precision at a confidence threshold of 0.5. This is a better result than previous attempts using only RGB or infrared images have reported, [2, 21, 22, 23]. Although this result must be seen in perspective with the lower localisation quality demand of a grid output compared to bounding box outputs, these high performance results show that automatic detection of sheep in dual RGB and infrared UAV images can be accomplished with a high quality using deep learning. Nonetheless, there are some limitations to the results and methodology of this project, which will be discussed in this section.

The scope of this thesis is limited to analysing CNN model design decisions for automatic sheep detection, which is just one part of the total system required for this to be used commercially in the field. More work and research is required to find out how farmers would use such a product; which hardware is most suitable for the desired use and cost requirements; how to efficiently and reliably transfer data between the camera, image processing device and graphical user interface; how to best design a user interface that easily allows the farmers to use the system; and which laws, regulations and ethical questions relating to privacy and safety must be considered.

## Result Dependencies

The results of such a system is very dependent on the available hardware. The two modules of the envisioned solution described in section 3.1 have different requirements to the UAV. The M2ED that is used in this project would be suitable for the real-time module since it is a small, compact and relatively affordable device that can easily be transported out in the field. However, it is not suitable for mapping a larger area since the low resolution of its infrared camera limits how high it can fly and the battery is limited to maximum 31 minutes. Purchasing a UAV with higher endurance and higher quality camera that is required to map a larger area, will likely be rather expensive. This may limit the feasibility of practically implementing a system that involves surveying large areas with a dual infrared and RGB camera.

Inference time considerations are strongly dependent on speed, memory, and allocation of the available hardware. Although inference times reported in this thesis are within the requirements of the system as explained in section (3.2), these results would look different if processing was done on the user's mobile phone or on some small machine mounted on the UAV.

The results of the model are also dependent on the location where data is captured and the properties of the sheep and environment in that area. The models in this thesis are trained on data captured in Storlidalen in Oppdal, where sheep and environmental properties potentially vary from data collected in another location. Therefore, the same results will not necessarily be obtained if the model is applied to data from another location (as shown by the inferior performance on the T2 dataset). If such a product was to be used commercially, a way to solve this issue is to let the users themselves submit data to be used for training the network. This way the trained model would become familiar with its target sheep and environments.

## Model Design and Evaluation Choices

The choice to produce grid output instead of bounding boxes was made to save processing time and to simplify the task in expectation that this would give better detection results. However, the sacrifice of localisation quality means the user gets less precise results, which are more difficult to understand. Moreover, bounding boxes are beneficial because they not only inform the user of where the sheep are but also the exact number of sheep that have been detected. Future research could investigate whether there are sufficient performance gains to justify this simplified approach compared to a system that outputs bounding boxes.

The choice of average precision as a performance measure can also be questioned because the grid output produced by the network is more similar to the pixel mask output of a semantic segmentation network than the bounding box output of an object detection network. In retrospect, it could be more appropriate to consider another metric such as the F1 score, which commonly used to evaluate segmentation performance.



## 7 Conclusion and Future Work

In this thesis, deep learning model architectures that are custom tailored to the task of accurate and time-efficient automatic detection of sheep in infrared and RGB images have been proposed. These architectures have been implemented and tested using a range of configurations with varying input types, model designs and image resolutions to find a set of optimal solutions that give the best average precision vs. inference time trade-off. The dataset used for training and main evaluation the sheep detection models are a set of RGB and infrared image pairs of sheep grazing in a range of free and fenced environments, captured in Storlidalen valley in Oppdal.

Findings of the research confirms the hypothesis laid out in previous work on automatic sheep detection in UAV imagery that fusion of RGB and infrared data yields better performance results than using the two inputs separately. The fusion models also give a better average precision vs. inference time trade-off than the single input RGB models but are unable to match the fast processing speed of the low-resolution infrared models. By combining information from RGB and infrared images, the fusion model can take advantage of body heat information in the infrared image as well as texture, shape and colour details of the RGB image. Observations of sample predictions show that the two input data types often complement each other by endorsing each other's correct predictions and opposing each other false positive predictions.

Fusion depth, model complexity and RGB resolution are also factors that contribute to average precision and inference-time results. Results show that fusion after the fourth convolutional layer of the ResNet or ResNeXt backbone CNN will give the best results for this network design and use case. Increasing model complexity and RGB resolution both cause gains to average precision, however these adjustments come at significant costs to inference time.

This thesis distinguishes itself from previous work by the decision to output probabilities of sheep presence in a fixed grid area of the image instead of bounding boxes for each individual sheep. This simplification of the task can be made because such strict localisation quality as bounding boxes is not required for the task of finding the sheep. On the other hand, this sacrifice to precision makes the network results more difficult to interpret and means that the model cannot be used for counting sheep.

It should be noted that results are dependent on the hardware used and the location that data was captured in. Repeating the experiments with another UAV and computer with images from another location may produce different results.

Nonetheless, the high performance results achieved by the sheep detection models trained in this thesis shows that automatic detection of sheep in dual RGB and infrared UAV images can be accomplished with high quality and acceptable processing times using a deep learning approach. Integration of such a model into an easy to use commercial system could greatly aid farmers with locating their sheep in larger areas of challenging terrain. This will save valuable time and physical effort that otherwise would be involved in a manual search.

### 7.1 Future Work

Future work to address the limitations and build on work in this thesis involves collecting more data, taking a deeper look at processes that can be bottlenecking the processing time, consider alternative neural network architectures and systematically evaluating other important factors that contribute to model performance.

Despite this thesis featuring a larger dataset than previous research on this topic, there was not enough independent data for a separate test dataset and the validation dataset used to evaluate the model only consisted of data from three separate sessions. This is a limitation to the validity of the results. Future work should therefore aim to collect more data and use an independent and representative test dataset to evaluate results.

The decision to output grid probabilities rather than bounding boxes simplifies the architecture and

reduces processing time at the expense of a lower localisation quality. Future research could investigate whether there are sufficient performance gains to justify this simplified approach compared to a system that outputs bounding boxes. Currently there are a great number of object detection architectures that achieve state of the art detection results for bounding box outputs and new architectures are constantly emerging. By integrating the backbone fusion model used in this thesis into an existing object detection toolbox, new architectures can easily be implemented and adapted to benefit from fusion of RGB and infrared input data.

In addition, further work to identify which parts of the inference process are contributing the most to inference time would give useful insights to help better understand inference time results and ways to reduce this metric within the available hardware constraints.

Other factors that could be interesting to investigate further are UAV flight height and the use of MSX style infrared images. Finding a good trade-off between flight height and performance would be useful for finding out how high images can be captured from while maintaining an acceptable performance. Moreover, it could be considered to include flight height as input to the model as this can give the network a clue to how large the sheep in the image can be expected to be and thus eliminate candidates that are outside a reasonable size range. MSX images are produced in real time by the FLIR imaging technology in the camera UAV used for this thesis. MSX images have the same low-resolution as the regular infrared images but contain embedded edge features of the RGB images. As a result, processing of these images could be a good alternative to the fusion model when fast processing time is a priority.

## Bibliography

- [1] Kristiansen, N. 2019. Hvor plagsomt er det for sauene å gå med ei bjelle rundt halsen hele sommeren? URL: <https://forskning.no/dyreverden-husdyr-landbruk/hvor-plagsomt-er-det-for-sauen-a-ga-med-ei-bjelle-rundt-halsen-hele-sommeren/1360767>.
- [2] Even Arneberg Rognlien, T. Q. T. & Hvasshovd, S.-O. 2018. Detecting location of free range sheep using unmanned aerial vehicles and forward looking infrared images.
- [3] Saha, S. 2018. A comprehensive guide to convolutional neural networks — the eli5 way. URL: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>.
- [4] Karim, R. 2019. Illustrated: 10 cnn architectures. URL: <https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d>.
- [5] Bronshtein, A. 2017. Common loss functions in machine learning. URL: <https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6>.
- [6] Hassan, A. 2019. Transfer learning from rgb to multi-band imagery. URL: <https://www.azavea.com/blog/2019/08/30/transfer-learning-from-rgb-to-multi-band-imagery/>.
- [7] Ophoff, T., Van Beeck, K., & Goedemé, T. 02 2019. Exploring rgb+depth fusion for real-time object detection. *Sensors*, 19, 866. doi:10.3390/s19040866.
- [8] og Gjeit (NSG), N. S. 2019. Sauerasene i norge. URL: <http://www.nsg.no/saueraser-i-norge/category719.html>.
- [9] EOL. 2020. Sheep. URL: <https://eol.org/pages/311906/data>.
- [10] He, K., Zhang, X., Ren, S., & Sun, J. 2015. Deep residual learning for image recognition. arXiv:1512.03385.
- [11] norske leksikon, S. 2019. Sauehold i norge. URL: <https://snl.no/sau>.
- [12] Smartbjella. 2019. Smartbjella. URL: <https://smartbjella.no/>.
- [13] findmy. 2019. findmy. URL: <https://www.findmy.no/>.
- [14] Telespor. 2019. Elektronisk overvåking av husdyr. URL: <https://telespor.no/>.
- [15] Anne-Cath. Grimstad, N. S. o. G. 2017. Tilsyn med drone: Rimelig og effektivt. *Sau og Geit*, Sau og Geit Nr.4/2017. doi:<http://dx.doi.org/10.1002/andp.19053221004>.
- [16] Redmon, J. & Farhadi, A. 2018. Yolov3: An incremental improvement. arXiv:1804.02767.
- [17] Cai, Z. & Vasconcelos, N. 2017. Cascade r-cnn: Delving into high quality object detection. arXiv:1712.00726.
- [18] Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P. 2017. Focal loss for dense object detection. arXiv:1708.02002.
- [19] Pang, J., Chen, K., Shi, J., Feng, H., Ouyang, W., & Lin, D. 2019. Libra r-cnn: Towards balanced learning for object detection. arXiv:1904.02701.
- [20] Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., & Dollár, P. 2014. Microsoft coco: Common objects in context. arXiv:1405.0312.

- [21] Marit Gjøstøl Ytterland, T. K. E. W. & Hvasshovd, S.-O. 2019. Retrieval of sheep using unmanned aerial vehicle.
- [22] Muribø, J. H. & Hvasshovd, S.-O. 2019. Locating sheep with yolov3.
- [23] Johannessen, K. M. Detecting sheep in drone images by deep learning. Project report in TBA4560, Department of Civil and Environmental Engineering, NTNU – Norwegian University of Science and Technology, Dec. 2019.
- [24] Nortura. 2019. Nortura hjemmeside. URL: <http://www.nortura.no/>.
- [25] Nortura. 2019. Temahefte - utmarksbeite til sau. URL: [https://www.geno.no/globalassets/geno-sa/02\\_dokumenter/02\\_aktiviteter/06\\_husdyrtreff/oppgave-sau/2020/utmarksbeite\\_sau\\_web.pdf](https://www.geno.no/globalassets/geno-sa/02_dokumenter/02_aktiviteter/06_husdyrtreff/oppgave-sau/2020/utmarksbeite_sau_web.pdf).
- [26] Hvasshovd, S.-O. 2019. Droner og sau og litt til !! anvendelser og muligheter. URL: <https://www.fylkesmannen.no/contentassets/cbf122460efa4e37a051c17c07fade0d/droner-buskerud-2017.pdf>.
- [27] Burke, C., Rashman, M., Wich, S., Symons, A., Theron, C., & Longmore, S. 2019. Optimizing observing strategies for monitoring animals using drone-mounted thermal infrared cameras. *International Journal of Remote Sensing*, 40(2), 439–467. URL: <https://doi.org/10.1080/01431161.2018.1558372>, arXiv:<https://doi.org/10.1080/01431161.2018.1558372>, doi: 10.1080/01431161.2018.1558372.
- [28] Mahony, N. O., Campbell, S., Carvalho, A., Harapanahalli, S., Velasco-Hernandez, G., Krpalkova, L., Riordan, D., & Walsh, J. 2019. Deep learning vs. traditional computer vision. arXiv:1910.13796.
- [29] MathWorks. 2019. What is object detection? URL: <https://se.mathworks.com/discovery/object-detection.html>.
- [30] Zou, Z., Shi, Z., Guo, Y., & Ye, J. 2019. Object detection in 20 years: A survey. arXiv:1905.05055.
- [31] Pancast. 2019. Object detection: What is it and how is it useful? URL: <https://www.panacast.com/object-detection-what-is-it-and-how-is-it-useful/>.
- [32] Everingham, M., Gool, L. V., Williams, C. K. I., Winn, J. M., & Zisserman, A. 2009. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88, 303–338.
- [33] Huang, T. S. 1996. Computer vision: Evolution and promise. URL: <https://cds.cern.ch/record/400313/files/p21.pdf>.
- [34] Khan, A., Sohail, A., Zahoora, U., & Qureshi, A. S. 2019. A survey of the recent architectures of deep convolutional neural networks. arXiv:1901.06032.
- [35] Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Xu, J., Zhang, Z., Cheng, D., Zhu, C., Cheng, T., Zhao, Q., Li, B., Lu, X., Zhu, R., Wu, Y., Dai, J., Wang, J., Shi, J., Ouyang, W., Loy, C. C., & Lin, D. 2019. Mmdetection: Open mmlab detection toolbox and benchmark. arXiv:1906.07155.
- [36] He, K., Gkioxari, G., Dollár, P., & Girshick, R. B. 2017. Mask R-CNN. *CoRR*, abs/1703.06870. URL: <http://arxiv.org/abs/1703.06870>, arXiv:1703.06870.
- [37] Nwankpa, C., Ijomah, W., Gachagan, A., & Marshall, S. 2018. Activation functions: Comparison of trends in practice and research for deep learning. arXiv:1811.03378.
- [38] Ioffe, S. & Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv:1502.03167.
- [39] Amidi, A. & Amidi, S. 2017. Convolutional neural networks cheatsheet. URL: <https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks>.

- [40] Wojna, Z., Ferrari, V., Guadarrama, S., Silberman, N., Chen, L.-C., Fathi, A., & Uijlings, J. 2017. The devil is in the decoder: Classification, regression and gans. [arXiv:1707.05847](https://arxiv.org/abs/1707.05847).
- [41] Lane, T. 2018. Transposed convolutions explained with... ms excel! URL: <https://medium.com/apache-mxnet/transposed-convolutions-explained-with-ms-excel-52d13030c7e8>.
- [42] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.
- [43] Krizhevsky, A., Sutskever, I., & Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, Pereira, F., Burges, C. J. C., Bottou, L., & Weinberger, K. Q., eds, 1097–1105. Curran Associates, Inc. URL: <http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>.
- [44] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. 2014. Going deeper with convolutions. [arXiv:1409.4842](https://arxiv.org/abs/1409.4842).
- [45] Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K. 2016. Aggregated residual transformations for deep neural networks. [arXiv:1611.05431](https://arxiv.org/abs/1611.05431).
- [46] Glossary, M. L. 2020. Machine learning glossary. URL: <https://ml-cheatsheet.readthedocs.io/en/latest/index.html>.
- [47] Andy. 2020. Stochastic gradient descent – mini-batch and more. URL: <https://adventuresinmachinelearning.com/stochastic-gradient-descent/>.
- [48] Parmar, R. 2018. Common loss functions in machine learning. URL: <https://towardsdatascience.com/common-loss-functions-in-machine-learning-46af0ffc4d23>.
- [49] Edu, S. 2019. Cs231n convolutional neural networks for visual recognition. URL: <http://cs231n.github.io/>.
- [50] Hassan, A. 2019. Transfer learning from rgb to multi-band imagery. URL: <https://www.azavea.com/blog/2019/08/30/transfer-learning-from-rgb-to-multi-band-imagery/>.
- [51] Perez, L. & Wang, J. 2017. The effectiveness of data augmentation in image classification using deep learning. [arXiv:1712.04621](https://arxiv.org/abs/1712.04621).
- [52] Brownlee, J. 2019. How to configure the learning rate when training deep learning neural networks. URL: <https://machinelearningmastery.com/learning-rate-for-deep-learning-neural-networks/>.
- [53] Luo, L., Xiong, Y., Liu, Y., & Sun, X. 2019. Adaptive gradient methods with dynamic bound of learning rate. [arXiv:1902.09843](https://arxiv.org/abs/1902.09843).
- [54] Masters, D. & Luschi, C. 2018. Revisiting small batch training for deep neural networks. [arXiv:1804.07612](https://arxiv.org/abs/1804.07612).
- [55] Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., & He, K. 2017. Accurate, large minibatch sgd: Training imagenet in 1 hour. [arXiv:1706.02677](https://arxiv.org/abs/1706.02677).
- [56] paperswithcode.com. 2020. Object detection on coco test-dev. URL: <https://paperswithcode.com/sota/object-detection-on-coco>.
- [57] Wu, X., Sahoo, D., & Hoi, S. C. H. 2019. Recent advances in deep learning for object detection. [arXiv:1908.03673](https://arxiv.org/abs/1908.03673).
- [58] Girshick, R., Donahue, J., Darrell, T., & Malik, J. 2013. Rich feature hierarchies for accurate object detection and semantic segmentation. [arXiv:1311.2524](https://arxiv.org/abs/1311.2524).
- [59] pascal VOC. 2012. Visual object classes challenge 2012 (voc2012). URL: <http://host.robots.ox.ac.uk/pascal/VOC/voc2012/>.

- [60] Girshick, R. 2015. Fast r-cnn. arXiv:1504.08083.
- [61] Ren, S., He, K., Girshick, R., & Sun, J. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. arXiv:1506.01497.
- [62] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. 2015. You only look once: Unified, real-time object detection. arXiv:1506.02640.
- [63] Redmon, J. & Farhadi, A. 2016. Yolo9000: Better, faster, stronger. arXiv:1612.08242.
- [64] Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. 2016. Feature pyramid networks for object detection. arXiv:1612.03144.
- [65] Tian, Z., Shen, C., Chen, H., & He, T. 2019. Fcos: Fully convolutional one-stage object detection. arXiv:1904.01355.
- [66] ISPRS. 2020. 2d semantic labeling contest - potsdam. URL: <http://www2.isprs.org/commissions/comm3/wg4/2d-sem-label-potsdam.html>.
- [67] Ophoff, T., Goedemé, T., & Van Beeck, K. 11 2018. Improving real-time pedestrian detectors with rgb+depth fusion. doi:10.1109/AVSS.2018.8639110.
- [68] Hong Tian, Enhai Liu, R. Z. Y. H. & Zuo, D. 2016. A decoupled calibration method for camera intrinsic parameters and distortion coefficients.
- [69] Zhang, Z. 2016. Camera parameters (intrinsic, extrinsic). URL: [https://link.springer.com/referenceworkentry/10.1007%2F978-0-387-31439-6\\_152](https://link.springer.com/referenceworkentry/10.1007%2F978-0-387-31439-6_152).
- [70] openCV. Camera calibration and 3d reconstruction (online). 2019. Accessed: 2019-24-09. URL: [https://opencv-python-tutroals.readthedocs.io/en/latest/py\\_tutorials/py\\_calib3d/py\\_calibration/py\\_calibration.html](https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_calib3d/py_calibration/py_calibration.html).
- [71] Brown, D. H. 1966. Decentering distortion of lenses.
- [72] de Villiers, J. N. P., Leuschner, F. W., & Geldenhuys, R. 2008. Centi-pixel accurate real-time inverse distortion correction. In *International Symposium on Optomechatronic Technologies*.
- [73] Zhang, Z. 2000. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11), 1330–1334.
- [74] Castaldo, F. & Palmieri, F. A. 2013. Camera system: pinhole model, calibration and reconstruction. URL: <http://www.francescocastaldo.org/tutorials/camera.pdf>.
- [75] Gentle, J. 2007. *Matrix Algebra: Theory, Computations, and Applications in Statistics*. Springer Texts in Statistics. Springer New York. URL: <https://books.google.no/books?id=PDjIV0iWa2cC>.
- [76] DroneZon. 2019. Top mavic 2 enterprise review and faqs – thermal, dual spotlight, loudspeaker and beacons mounts. URL: <https://www.dronezon.com/drone-reviews/mavic-2-enterprise-review-with-spotlights-loudspeaker-beacon-faqs/>.
- [77] Labelbox. 2019. Labelbox homepage. URL: <https://labelbox.com/>.
- [78] FLIR. 2019. What is msx. URL: <https://www.flir.com/discover/professional-tools/what-is-msx/>.
- [79] scikit image. 2019. scikit-image documentation for transform module. URL: <https://scikit-image.org/docs/dev/api/skimage.transform>.
- [80] PyTorch. 2019. From research to production. URL: <https://pytorch.org/>.
- [81] He, H. 2019. The state of machine learning frameworks in 2019. URL: <https://thegradients.pub/state-of-ml-frameworks-2019-pytorch-dominates-research-tensorflow-dominates-industry/>.

- 
- [82] Hukkelås, H. 2020. Tdt4265-startercode. URL: <https://github.com/hukkelas/TDT4265-StarterCode>.
- [83] A. Buslaev, A. Parinov, E. K. V. I. I. & Kalinin, A. A. 2018. Albumentations: fast and flexible image augmentations. *ArXiv e-prints*. arXiv:1809.06839.
- [84] learn, S. 2020. sklearn metrics average\_precision\_score(). URL: [https://scikit-learn.org/stable/modules/generated/sklearn.metrics.average\\_precision\\_score.html](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.average_precision_score.html).

## A Appendix



## A.1 Overview of Sections Adapted or Included from Specialisation Project

**Section 2.1 Sheep Grazing and Roundup:** This section is largely identical to the corresponding section in the specialisation project. The traditional bell was added to the list of existing technologies.

**Section 2.3 Object Detection:** This section is identical to the corresponding section in the specialisation project research as the definition and common metrics for object detection remain unchanged.

**Section 2.4 Deep Learning:** The majority of this section is original, however subsection 2.4.5 is highly adapted from the corresponding section in the specialisation project since historically influential architectures for deep learning-based object detection remain relevant and unchanged.

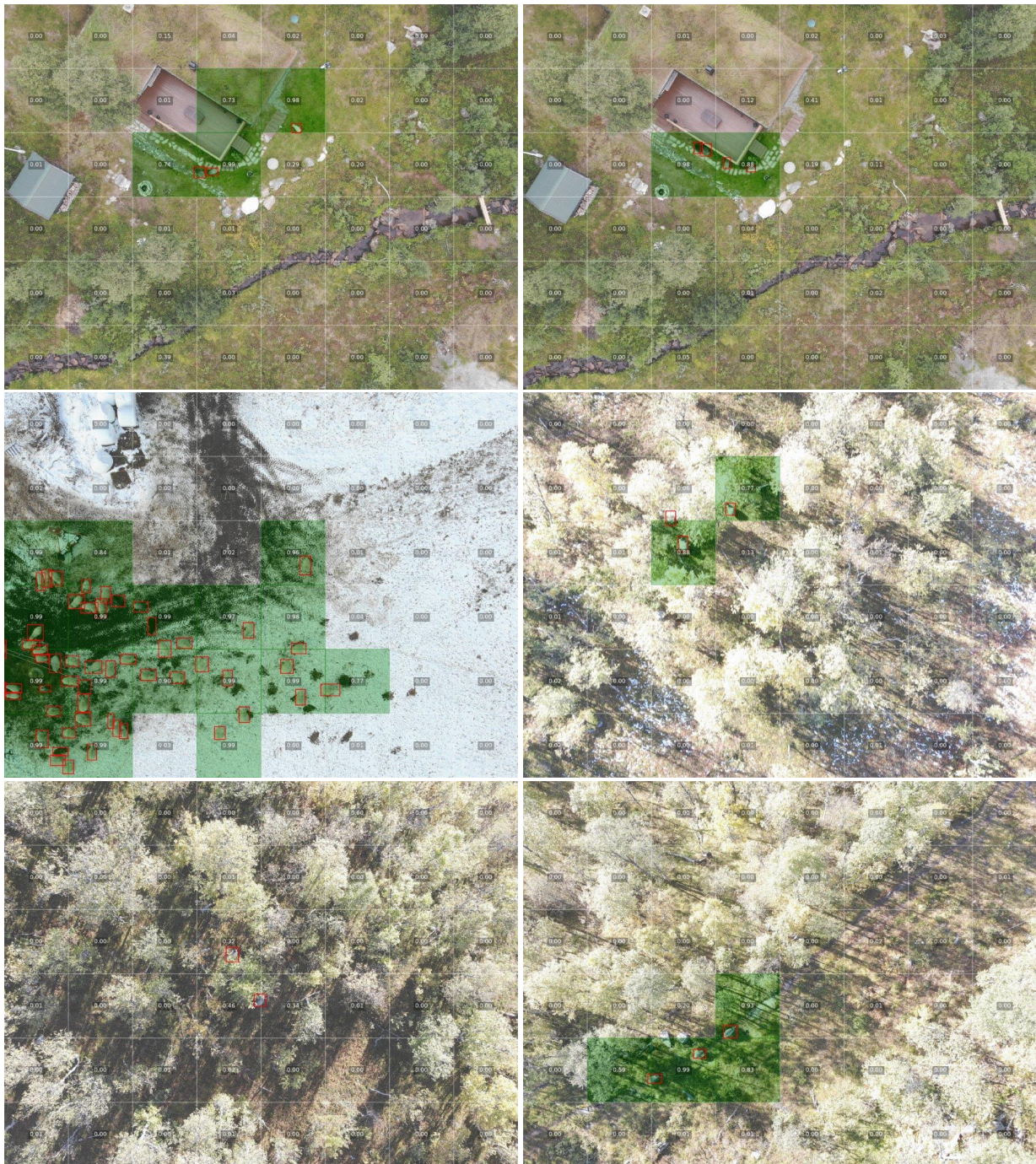
**Section 3.2 Requirements:** Requirements 4 to 7 are the same but requirements 1 to 3 have changed to various degrees. Most noticeably, processing time requirements are discussed in detail.


**Section 3.4 Hardware Constraints:** The same drone and computer are used so this text is identical.


## A.2 Sample of Labelled Images In Training Dataset



### A.3 Sample of Images and Predictions on Validation Dataset



 Ground Truth  
Bounding Box

 Grid Confidence >  
0.5

X.XX

Predicted Sheep  
Confidence Score

## A.4 Sample of Images and Predictions on T1 Datasets



Ground Truth  
Bounding Box



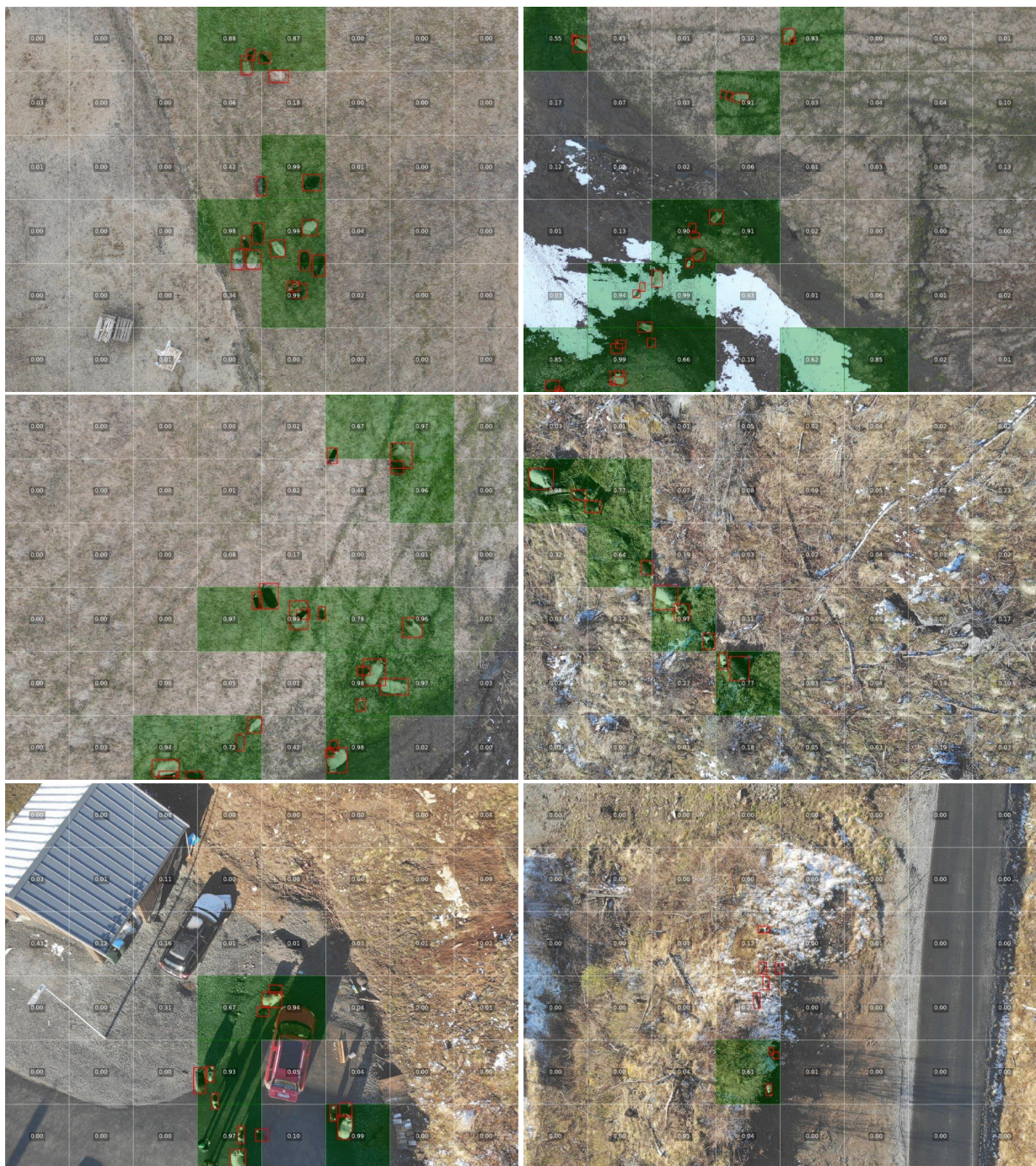
Grid Confidence >  
0.5





x.xx

Predicted Sheep  
Confidence Score

## A.5 Sample of Images and Predictions on T2 Datasets



 Ground Truth Bounding Box

 Grid Confidence > 0.5

X.XX

Predicted Sheep Confidence Score

## A.6 Results of All Models

Model	Fusion Depth	RGB Size	Infrared Size	Average precision				Inference Time (s)
				Training	Validation	T1	T2	
I_r101	-	-	64	0.882	0.840	0.620	0.339	0.100
I_r18	-	-	64	0.876	0.711	0.707	0.448	0.101
I_r18	-	-	64	0.884	0.770	0.703	0.423	0.100
I_r18	-	-	96	0.889	0.761	0.735	0.465	0.102
I_r18	-	-	96	0.808	0.696	0.682	0.487	<b>0.098</b>
I_r18	-	-	128	0.858	0.714	0.720	0.498	0.099
I_r18	-	-	128	0.842	0.761	0.726	0.482	0.103
I_r18	-	-	160	0.863	0.740	0.671	0.467	0.103
I_r18	-	-	160	0.867	0.777	0.665	0.484	0.103
I_r50	-	-	64	0.893	0.833	0.585	0.365	0.098
I_rx50	-	-	64	0.909	0.808	0.770	0.440	0.109
RGB_r18	-	128	-	0.815	0.404	0.766	0.382	0.147
RGB_r18	-	128	-	0.878	0.416	0.824	0.382	0.154
RGB_r18	-	160	-	0.953	0.520	0.843	0.573	0.153
RGB_r18	-	160	-	0.873	0.547	0.828	0.394	0.155
RGB_r18	-	256	-	0.959	0.713	0.875	0.634	0.155
RGB_r18	-	256	-	0.906	0.688	0.850	0.528	0.152
RGB_r18	-	320	-	0.962	0.719	0.889	0.698	0.260
RGB_r18	-	320	-	0.819	0.691	0.745	0.473	0.255
RGB_r18	-	384	-	0.923	0.752	0.831	0.601	0.255
RGB_r18	-	384	-	0.925	0.756	0.843	0.611	0.258
RGB_r18	-	512	-	0.921	0.794	0.819	0.623	0.265
RGB_r18	-	512	-	0.902	0.776	0.811	0.627	0.261
RGB_r18	-	640	-	0.935	0.796	0.810	0.683	0.268
RGB_r18	-	640	-	0.983	0.850	0.920	0.775	0.266
RGB_r18	-	764	-	0.944	0.794	0.874	0.727	0.275
RGB_r18	-	768	-	0.940	0.836	0.839	0.718	0.274
RGB_r18	-	896	-	0.943	0.810	0.850	0.707	0.283
RGB_r18	-	896	-	0.963	0.827	0.902	0.748	0.275
RGB_r18	-	1024	-	0.919	0.811	0.839	0.690	0.287
RGB_r18	-	1024	-	0.932	0.824	0.847	0.726	0.284
RGB_r18	-	1152	-	0.894	0.802	0.785	0.675	0.294
RGB_r18	-	1152	-	0.890	0.783	0.762	0.689	0.299
RGB_r18	-	1280	-	0.835	0.784	0.692	0.650	0.312
RGB_r18	-	1280	-	0.860	0.777	0.744	0.641	0.309
RGB_r50	-	1024	-	0.936	0.871	0.852	0.801	0.380
RGB_rx50	-	1024	-	0.970	0.899	0.918	0.818	0.477
RGB+I_r18	3	512	64	0.985	0.889	0.933	0.676	0.339
RGB+I_r18	3	640	160	0.988	0.894	0.944	0.731	0.337
RGB+I_r18	3	768	96	0.986	0.925	0.945	0.723	0.349
RGB+I_r18	3	1024	64	0.986	0.932	0.926	0.708	0.360
RGB+I_r18	4	128	64	0.973	0.832	0.786	0.385	0.232
RGB+I_r18	4	256	64	0.983	0.879	0.912	0.652	0.233
RGB+I_r18	4	512	64	0.985	0.908	0.924	0.708	0.329
RGB+I_r18	4	640	160	0.993	0.913	0.928	0.796	0.340
RGB+I_r18	4	768	96	0.990	0.920	0.938	0.771	0.340
RGB+I_r18	4	1024	64	0.991	0.942	0.939	0.804	0.355
RGB+I_r18	5	512	64	0.991	0.871	0.923	0.702	0.334
RGB+I_r18	5	640	160	0.984	0.897	0.905	<b>0.851</b>	0.335
RGB+I_r18	5	768	96	0.986	0.900	0.906	0.811	0.340
RGB+I_r18	5	1024	64	0.966	0.893	0.910	0.732	0.356
RGB+I_r18	6	512	64	0.983	0.892	0.879	0.726	0.333
RGB+I_r18	6	640	160	0.987	0.886	0.926	0.786	0.341
RGB+I_r18	6	768	96	0.978	0.908	0.889	0.753	0.340
RGB+I_r18	6	1024	64	0.964	0.890	0.872	0.700	0.355
RGB+I_r50	4	1024	64	0.990	0.959	0.940	0.810	0.465
RGB+I_rx50	4	1024	64	<b>0.993</b>	<b>0.963</b>	<b>0.945</b>	0.823	0.586

Table 16: Results recorded for all trained models.

