Wildlife Protection through Aerial Drone Surveillance

by

Isobel Lees, 1003912484
Tiffany Yau, 1004191867
Helen Wang, 1004030185
Aditi Maheshwari, 1004175420

April 2022

ROB498 Project Report
Division of Engineering Science
University of Toronto
Canada

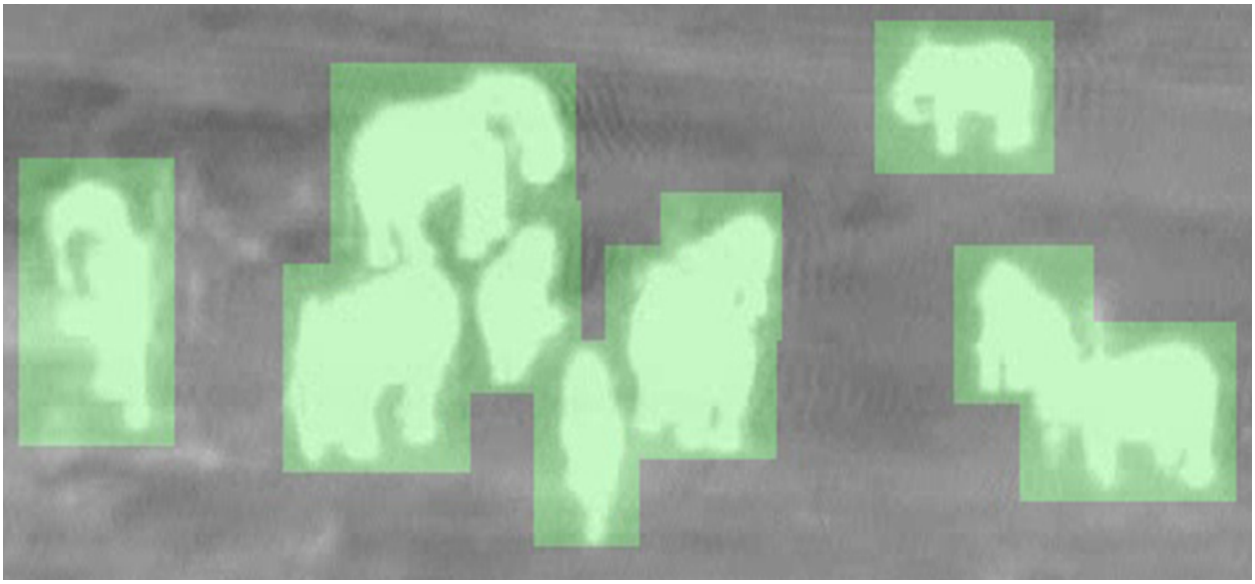

Table of Contents

List of Figures

List of Tables

1 Project Overview

Wildlife endangerment is a serious issue facing the world today. The rapid growth of humanity has destroyed countless habitats through deforestation, pollution, invasive species, warming temperatures, and extreme weather (i.e. wildfires) [1, 2]. Moreover, there is a serious threat from poachers motivated by the enormous profits from selling animal parts to criminal organisations [3]. For example, over 50,000 elephants are killed each year for their ivory, while rhino horns sell for over 50,000 euros [3]. This problem also exists in Canada with the trade of illegal furs [4, 5].

There are solutions to reduce poaching, including the development of synthetic alternatives (i.e. fake ivory), ground patrols, and video surveillance [3]. One bottleneck with surveillance is that there is a lot of area to cover, and it is difficult to maintain cameras in remote areas. As a result, unmanned aerial vehicles (UAVs) equipped with imaging technology that can perform object detection and send warning signals back to base if they detect a problem (such as poachers or habitat destruction) could be very beneficial to wildlife protection efforts [6]. Hence, we saw an opportunity to assist by building a detection system that can recognize animals in thermal infrared aerial images. The final product would be useful to multiple stakeholders, including governments, NGOs and national park services across the world looking to monitor poaching activities.

We investigated multiple options, and performed several experiments before settling on our solution. Our final design is based on the You Only Learn One Representation (YOLOR) [7] neural network architecture trained to detect two classes of animals found in the Benchmarking Infrared Dataset for Surveillance with Aerial Intelligence (BIRDSAI) [8]. The success of the solution was measured through its percentage of correct identifications and speed. The final prototype exceeded our design requirement with a mean average precision (mAP) of 38.2% and is well within the hardware constraints of the GPU assumed to be available on the UAV.

2 Background

All objects with a temperature above 0 K will radiate heat in the form of electromagnetic waves. Unless the object is very hot (i.e. the sun), the wavelengths tend to be in the infrared (IR) range instead of the visible range, notably for typical body and environmental temperatures. Thermal cameras are essentially IR sensors that detect tiny differences in heat (as small as 0.01 °C) and create a human readable image [9].

While most existing algorithms for image detection are designed for RGB images, thermal data has several advantages. As we intend to monitor wildlife poaching, it is important to consider poachers' attempts to visually camouflage when hunting, either by hiding behind vegetation, wearing special clothing. In addition, many animals are nocturnal, and poaching activity can occur at night. Thermal cameras will work during the day as well as at night and can see objects under dynamic lighting, or objects with no visual contrast, unlike night vision and RGB cameras [9, 10]. Thermal infrared (TIR) data

also provides greater visibility in rain and other bad weather conditions [11, 12]. TIR cannot see through dense foliage or terrain but it is unlikely it will be necessary for our initial use-case. Furthermore, the images are captured from above so there are limited occlusions. Thus, thermal image data has unique potential for detection.

Moving away from the image capturing technology, the main focus of this project is building software that can analyze the TIR images and detect animals. Object detection refers to the unified task of object localization and classification in order to identify specific objects in given visual data. Specifically, this means placing a "bounding box" around objects of interest in the image, and then correctly identifying the object's class (e.g. cat, dog). Traditional machine learning object detection techniques employ methods including Haar Cascades [13] and Scale-Invariant Feature Transforms [14]. These techniques first detect and cluster features in the images, then perform cluster classification using logistic regression, colour value histograms, or random forests [15].

Current methods in object detection leverage recent developments in deep learning and GPU-accelerated computing. They have been shown to significantly outperform classical methods. Modern deep learning object detectors generally fall into two main categories: two-stage and single-stage. Two-stage object detectors first extract Regions of Interest (RoIs) (areas where there is a high probability of an object), and then perform classification and regression on the selected RoIs. Architectures that employ a two-stage structure include R-CNN, Fast-RCNN, and Faster-RCNN [16–18]. In contrast, single-stage detectors are initialized with images separated into grids and pre-defined anchor boxes. Anchor boxes are a group of boxes with particular sizes that are chosen to match the expected object dimensions [19]. They classify objects and regress bounding boxes using the anchors in a single network pass, skipping the RoI extraction process. A non-maximum suppression step then eliminates any overlap of predicted bounding boxes using a predefined Intersection over Union (IoU) threshold. Some examples of single-stage object detectors include the You Only Look Once (YOLOv2 to v5) family [20–23] and CenterNet [24]. Broadly speaking, single stage detectors are faster but can be less accurate (although modern ones can be quite accurate). For this project deep-learning based approaches that are capable of running inference onboard UAVs need to be chosen.

There are a number of deep learning-based algorithms for aerial detection with drones such as those given in the referenced papers [25–27]. Some focus specifically on animal detection in overhead images [28–35]. However, these detectors use RGB images and very few deal with TIR images [8, 36–39]. The dataset we used is BIRDSAI, a dataset for object detection containing sequences of aerial TIR images [8] (details in Section 4.1). Due to the limited research into animal detection on TIR images, we identified unexplored opportunity in this field.

3 Design Objectives

Due to limited time and resources, we scoped our design problem to focus on detection, which is the most important capability for our intended functionality. Our **original**

objectives were as follows (please find a detailed explanation of the metrics used to assess them in Section 3.1)

- Accurate detection of humans and various animals indicated by a $mAP \geq 40\%$
- Efficient performance onboard standard UAVs (solution should run at a minimum of 10 FPS and using $GFLOPs < 1330$ - reasoning described in Section 3.2.3)
- Robustness against size variance (standard deviation of mAP between different sized objects should be $< 30\%$) and robustness against noise, camera motion, and limited resolution (determined using mAP)

Throughout our design and experimentation process, we found that the most difficult objective to achieve was our target $mAP \geq 40\%$. This objective was based on the results reported in the BIRDSAI paper [8]. However, due to compute and time constraints, we trained primarily on a subset of the data, which contained only 5% of the images. We attempted to train with all the data, but it took approximately five hours per epoch. This was unreasonable considering our need for over 100 epochs per training experiment. Naturally, this leads to a model that is less able to generalize to data it has not seen before (i.e. the test set). More detailed information about the data subsets are provided in Section 4.1. Moreover, we noticed when inspecting the failure cases of our model that many objects are difficult to detect even with the human eye. We observed that it is extremely difficult to determine if an object is present and localize it, much less classify the object. This causes poorer performance. Therefore, we modified our objective to have a targeted $mAP \geq 35\%$.

In addition, due to the difficulty in generalizing, we determined that robustness against size variance should not be singled out as a secondary objective. Rather, we decided that it is more important to focus on optimizing the performance of the model holistically, as measured using the mAP across objects of *all* sizes. In reality, the need for size robustness can be overcome by fixing an altitude for the UAV at which animal size will be such that the detector performs reliably.

With these modifications, our **updated objectives** for our final design are as follows:

- Accurate detection of humans and various animals indicated by a $mAP \geq 35\%$
- Efficient performance onboard standard UAVs (solution should run at a minimum of 10 FPS and using $GFLOPs < 1330$ (reasoning described in Section 3.2.3)

3.1 Metrics

To verify, validate, and evaluate the design, we selected several key metrics commonly used to assess the performance (Section 3.1.1) and runtime (Section 3.1.2) of object detection methods.

3.1.1 Performance Metrics

One of the common metrics for object detection is mAP, which takes the mean of the average precision (AP) over all classes. $mAP_{0.5:0.95}$ can be computed using the 11-point interpolation method, which averages the precision values at 11 evenly spaced recall values (between 0.5 and 0.95 at 0.05 increments), where $Precision = \frac{TP}{TP+FP}$ and $Recall = \frac{TP}{TP+FN}$. TP is the number of true positives, and FP and FN are respectively the false positives and false negatives. $mAP_{0.5}$ is the precision value corresponding to $Recall = 0.5$.

To determine whether a detection is considered a false or true positive or negative, another metric known as Intersection-over-Union (IoU) is used. IoU measures the overlapping area between a predicted bounding box and the ground truth divided by the combined area. An IoU threshold is typically tuned to optimize mAP. For a detailed explanation, we refer readers to this paper by Padilla et al. [40].

3.1.2 Runtime Metrics

To assess runtime, we considered inference speed, measured in frames per second (FPS), including both the pre-processing (anchor box generation) and post-processing (non-maximum suppression) stages. We will also use the number of parameters and floating point operations per second (FLOPs) of the network to estimate the computation load required for a given model.

3.2 Constraints

3.2.1 Inherent to the design domain

Based on the drone type or TIR camera onboard, there would be limitations on altitude, hardware specifications, range, speed, camera resolution, data capacity and battery. These characteristics can severely limit the functionality of our solution. Moreover, altitude and perspective would affect the size and shape of animals in the TIR images, so our design should ideally work well for varying situations. It should be noted that TIR cameras would work well in different lighting and mild weather conditions. Dense foliage can block the UAV's aerial view, so it would be difficult to operate in thick forests. However, it should be able to see animals through scattered trees in grasslands without an issue.

3.2.2 Specific to our project

Since many poaching activities occur in remote areas or national parks, we will be dealing with insufficient internet service to transfer TIR data to the cloud. This means cloud computing will be limited. We will need to process the data onboard the drone and communicate results in real-time. We recognize that drones have limited compute, so our solution should be efficient. As a result, our design considers both performance and runtime metrics, discussed in Sections 3 and 3.1.

3.2.3 Hardware Assumptions

We referenced existing drones that are typically used for surveillance or are equipped with TIR cameras and made assumptions about the hardware on which our solution will be deployed. Specifically, we drew from the Skydio X2E Color/Thermal drone [41] and the senseFly eBee Geo [42], both of which are advertised for surveying. For compute specifications, we primarily refer to [41] and thus assume we have the NVIDIA Jetson Tegra X2 (TX2) [43] onboard the drone. This processor contains the following key components:

- 256-core NVIDIA Pascal™ GPU
- Dual-Core NVIDIA Denver 2 64-bit CPU
- Quad-Core ARM® Cortex®-A57 MPCore
- 8GB 128-bit LPDDR4 Memory (1866 MHx - 59.7 GB/s)

The Jetson TX2 permits a maximum of 1330 GFLOPs and also gives use a lower and upper bound on FPS, so our final design must meet these constraints [43]. As mentioned above, we assume the wireless service is only adequate for reporting outputs, so detection must be done onboard.

3.3 Concerns

Drones often pose privacy concerns when it comes to observing environments and gathering data. While the drones for monitoring animals will likely be placed in remote areas, they may need to fly around visitors or nearby residents. Additionally, security can be a major issue. Since the drones will be collecting animal locations, it is crucial to ensure this information is protected from poachers with malicious intent. Another concern is the impact of drones on birds [44,45]. If drones are used during the night, they can disrupt animals and may result in noise pollution.

4 Design Process and Prototype Overview

We began the design process by scoping the problem, developing objectives and constraints, and looking at the specifications of existing drones to gain an idea of what might be feasible. We also decided the objectives and metrics to evaluate our success. Then, we selected a suitable dataset to train our design solutions and prepared the data as required for the models we explored. The next step was to select and modify two deep learning-based 2D object detectors to output bounding boxes for each animal class using the aerial TIR images. We trained the models on the selected dataset and then verified, validated and evaluated the models. Variations of these models were also trained, and certain hyperparameters were altered to find the best possible model. After comparing all the different detection methods, we chose the one that that best fulfills our requirements as our final design. Figure 1 shows an overview of our design strategy. Details on dataset selection and preparation can be found in Section 4.1, model selection and design in Section 4.2, design optimization in Section 4.3, and model training and evaluation in Section 5. An overview

of our final design solution is presented in Section 6.

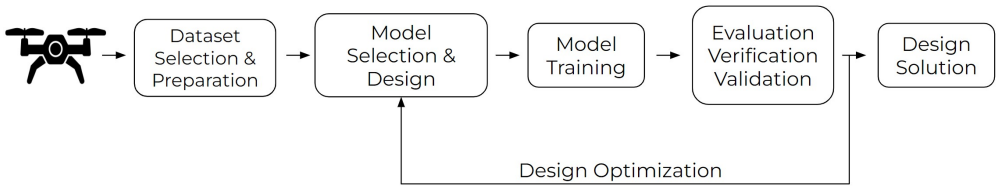

Figure 1: Overview of the design process.

4.1 Dataset Selection & Preparation

There are a number of publicly available 2D object detection datasets acquired from the aerial view using RGB cameras [46–48], but the TIR options were limited. The Benchmarking IR Dataset for Surveillance with Aerial Intelligence (BIRDSAI) [8] uniquely fits our use case by 1) capturing wildlife, 2) containing TIR images, and 3) having an aerial view. It is divided into 2 subsets: real and simulation. The real subset contains 48 real aerial TIR image sequences captured from protected areas throughout South Africa, Malawi, and Zimbabwe. The simulation subset contains 124 synthetic aerial TIR image sequences generated in a simulated African savanna environment in the AirSim-W [49] platform. The real training dataset, real testing dataset, and simulation training dataset are 2.1 GB, 1.6 GB, and 39.5 GB, respectively.

The provided annotations include 2D bounding boxes, class, species, occlusion, and noise annotations for all objects. The class label indicates whether or not the object is a human and the species label distinguishes between human, elephant, lion, giraffe, dog, crocodile, hippo, zebra, rhino, and unknown. Among these classes, giraffes and dogs are only found in the real data. Crocodiles, hippos, zebras, and rhinos are only found in the simulation data. The remaining animals can be found across both real and simulation data. Statistics of the bounding box count per class are shown in Table 1. Samples from both real and simulation subsets are shown for animals of varying sizes in Figure 2.

Table 1: **Dataset Statistics.** The Class Numbers correspond to the following animals: 0 - Humans, 1 - Elephants, 2 - Lion, 3 - Giraffe, 4 - Dog, 5 - Crocodile, 6 - Hippo, 7 - Zebra, 8 - Rhino, 9 - Unknown.

	Classes									
Split	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Train_Real	12525	42950	1020	9979	0	0	0	0	0	20684
Test_Real	22111	48037	351	3230	2715	0	0	0	0	2556
Train_Sim	48003	137692	1878	0	0	9098	26203	43893	2031	0

The BIRDSAI [8] dataset provided .csv files containing the annotations for each object at every timestep. To use this data for object detection, we converted the dataset to COCO [50] format. This streamlines experimentation with existing object detectors, which

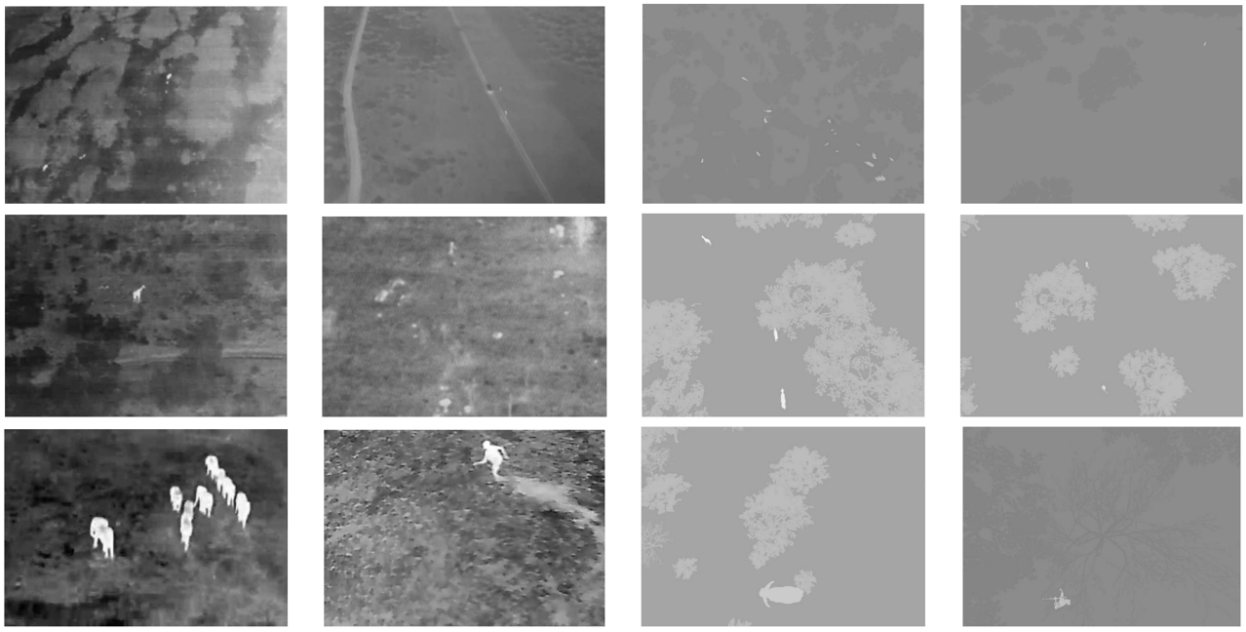

Figure 2: **Dataset image samples.** Samples of the real and synthetic images in BIRDSAI for small, medium, and large objects (from top to bottom). (a) and (b) show real images of animals and humans, respectively. (c) and (d) show synthetic images of animals and humans, respectively. Image sourced from [8].

commonly use the COCO dataset. In addition, many object detectors use a center-based bounding box representation of [center x, center y, width, height]. We converted the annotations of BIRDSAI from its original coordinates in [top left x, top left y, width, height] to the center-based representation.

4.2 Model Selection and Design

For our design, we modified existing 2D object detectors developed for three-channel RGB images to work on the TIR data in BIRDSAI. We reviewed recent and popular object detectors and created a list of those we wanted to explore:

- R-CNN [16]
- Fast R-CNN [17]
- Faster R-CNN [18]
- CenterNet [24]
- CenterNet2 [51]
- RetinaNet [52]
- EfficientDet - Scalable [53]
- YOLOv5 [23]
- YOLOR [7]
- PicoDet [54]

Note that YOLOR is not related to YOLOv5. They are developed by different authors.

In our selection process, we weighed the costs and benefits of object detectors with respect to performance and runtime. This was incorporated into a pairwise comparison chart shown in Table 2. We then narrowed down to the top three architectures – YOLOR [7], CenterNet2 [51], and YOLOv5 [23] – and focused on testing two of them. To provide some context, we give a more detailed summary of these top three methods and justify our selection of the top two models in the following paragraphs.

Table 2: **Pairwise Comparison.** Pairwise Comparison of Different Object Detector Architectures. 1 denotes the row architecture is better than the corresponding column architecture.

	YOLOR	YOLOv5	CenterNet2	CenterNet	EfficientDet	RetinaNet	RCNN	Fast RCNN	Faster RCNN
YOLOR		1	1	1	1	1	1	1	1
YOLOv5	0		0	1	1	1	1	1	1
CenterNet2	0	1		1	1	1	1	1	1
CenterNet	0	0	0		1	1	1	1	1
EfficientDet	0	0	0	0		1	1	1	1
RetinaNet	0	0	0	0	0		1	1	1
RCNN	0	0	0	0	0	0		0	0
Fast RCNN	0	0	0	0	0	0	1		0
Faster RCNN	0	0	0	0	0	0	1	1	

YOLOR stands for You Only Learn One Representation [7]. The method is aimed at learning a generalizable representation for input images that can be used for many different tasks. For the case of object detection, a CNN-based backbone is used for feature extraction, after which methods for implicit representation learning were applied to obtain the outputs. These features are then fed to detection heads to produce detection boxes, confidences, and classes.

CenterNet2 is a two-stage detector that, unlike other detectors, approaches detection from a probabilistic perspective [51]. In the first stage, it outputs bounding box proposals and corresponding object-vs-background likelihoods. Improving upon single-stage detectors in efficiency, it uses shared detection heads. In the second stage, it classifies proposals into specific object types using their likelihood.

Since YOLOR still achieves a comparable performance in comparison with CenterNet2, we chose to prioritize YOLOR due to the efficiency advantage. YOLOR learns a single object representation and shows effectiveness for multi-task learning. Onboard a UAV, this would benefit runtime of the overall system if the detection model is to be integrated with downstream tasks like tracking. Meanwhile, CenterNet2 does not provide this overall runtime benefit.

The "You Only Look Once" (YOLO) family of detectors is well-established. They are single-stage object detectors that divide images into a grid system with anchors centered in each grid cell [23]. For each anchor box, the model outputs potential detections. In YOLOv5, the data is inputted to a CSPDarknet53 [55] backbone for feature extraction [23]. Then, a PANet [56] neck is used to fuse features from different backbone levels together [23]. Finally, detection heads are used to output class, confidence score, location, and size.

In terms of detection accuracy and inference speed, YOLOv5 performs better than prior YOLO versions like YOLOv4 [23, 57]. However, direct comparisons have not been made between YOLOR and YOLOv5 or between CenterNet2 and YOLOv5. Since we do not have enough information to thoroughly compare YOLOv5 with YOLOR, which we have prioritized over CenterNet2, we chose to experiment with YOLOv5 as our second model.

4.3 Design Optimization

There are a number of ways to optimize object detectors to achieve our design objectives of performance, efficiency and robustness. For data preprocessing, we applied augmentation techniques, along with methods of label checking and data cleaning. Such data processing methods ensure the reliability of the training data, as well as maximizes the sample variety to facilitate generalization at inference time. In addition, we experimented with different model architectures to optimize for performance on TIR data, while aiming for a runtime practical on a standard UAV.

4.3.1 Data Amelioration

Data amelioration is the process through which we added channels to the model input by deriving key features from the BIRDSAI dataset's original one channel greyscale TIR images. We added one channel that is the output of a Canny edge detector [58] and another channel from the result of adaptive thresholding on the original image. We performed these image processing operations using the `OpenCV` Python library [59]. Figure 3 shows what each of the channels look like for two example images.

The results of these two image processing techniques and the original single-channel TIR image, totalling three input channels, were used as input to the object detectors. The use of three channels imitates an RGB image which contains different information in each channel. We expect that using three channels, each containing different information derived from the thermal data, will allow the object detector to perform better than when using the original single-channel TIR image in isolation.

The canny edge detector [58] starts with removing noise in the original image using a Gaussian filter. Next, a Sobel kernel is used to get the edge gradient magnitude and direction for each pixel. To obtain a binary image with edges, the Canny edge detector checks if each pixel is a local maximum in its neighbourhood in the direction of the gradient, which determines if the pixels form an edge. The last step consists of identifying if edges are actually edges using threshold values.

Image thresholding is the process by which pixel values of an image greater than a certain threshold are set to be 1 and values less than that threshold are set to be 0. We tried three types of thresholding: simple, adaptive, and Otsu's [60]. Simple thresholding applies the same threshold value to each pixel. This is not ideal for images with different pixel distributions, so adaptive thresholding determines the threshold for a pixel based

on a small region surrounding it. Otsu's thresholding [60] determines an optimal global threshold using the image histogram. Figure 4 shows the different types of thresholding for the first example image in Figure 3. We see that adaptive thresholding performs the best since the other approaches capture the terrain as features.

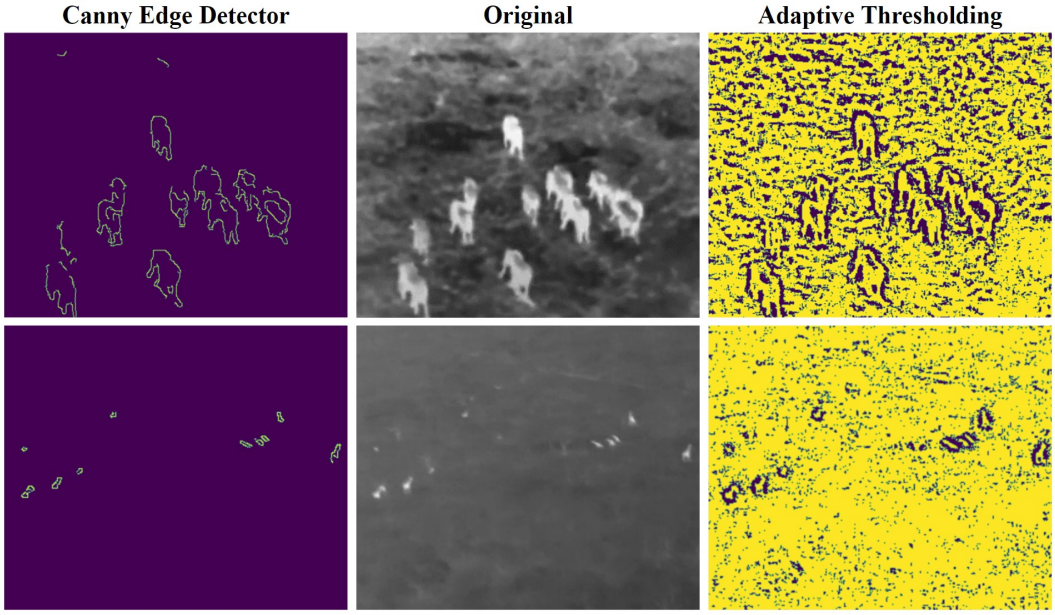

Figure 3: **Augmented Channel Visualizations.** Visualizations of the selected image processing techniques used for channel-wise data enhancement. Note that the thresholded images are shown in colour for better viewing, but they represent single (greyscale) channels.

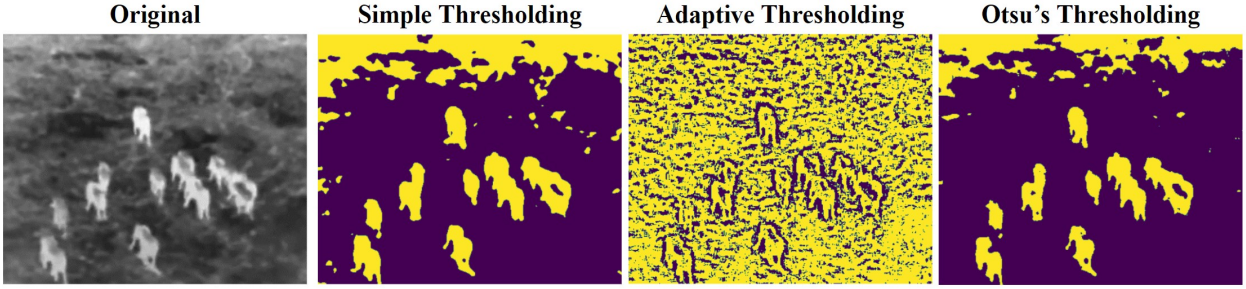

Figure 4: **Visualizations of thresholding methods.** Visualizations of the three types of thresholding we experimented with. Note that the thresholded images are shown in colour for better viewing, but they represent single (greyscale) channels.

4.3.2 Anchor Box Initialization

Object detection models are initialized with particular anchor box sizes that are used to form candidate bounding boxes [19]. The model predicts the offsets and other characteristics for the bounding boxes and these predictions are refined while the model trains.

Since anchors begin with fixed initial values, they must be representative of the object sizes present in the dataset, otherwise improper anchor sizes could lead to poor performance [19].

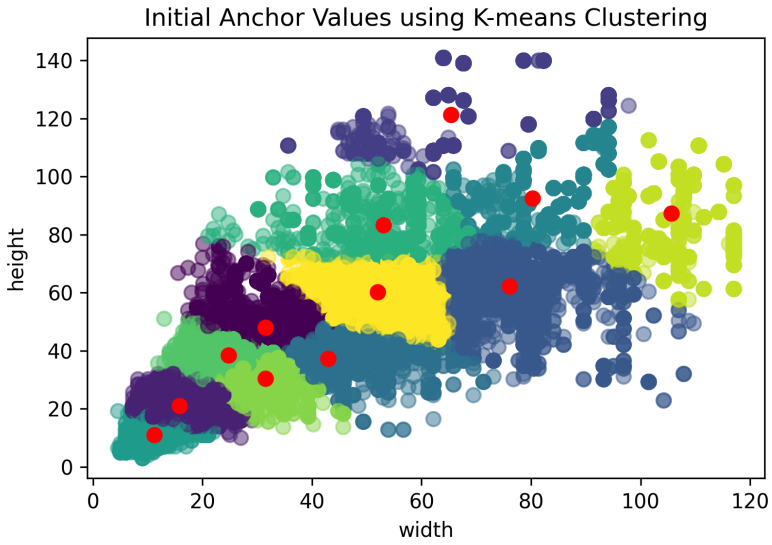

Figure 5: **K-means based anchor selection.** Anchor initialization using K-means clustering. The centroids representative of the anchor sizes are indicated by red dots for each cluster. Height and width are measured in pixels.

We initially derived anchor boxes using the most common aspect ratios and bounding box sizes from the training data; a modal approach. After reducing to 2 classes, we decided to try a K-means clustering algorithm to determine optimal anchor boxes. We arranged the width along the x-axis and height along the y-axis for each of the bounding boxes in the dataset as shown in Figure 5. The K-means clustering algorithm, implemented using the `scikit-learn` Python library, partitions these data points into 12 clusters. The centroids of these clusters were used at the initial anchor values. Figure 5 shows the centroids for each of the clusters in red.

4.3.3 Class Merging

The BIRDSAI Dataset was separated into a real (training and testing) and simulation (training) subsets. As mentioned in Section 4.1, neither the real data nor the simulation data contains all ten classes of animals. Inspecting the real data statistics further (refer to Table 1 in Section 4.1), it mostly consists of elephants (50%), humans (14%) and unknowns (24%). In fact, within the real dataset, the sum of the number of samples from all the other classes remains less than the total number of elephants. The simulation data follows similar trends, but as mentioned in Section 3, we were unable to train extensively using this subset due to compute limitations. When using the real subset in isolation, many of the classes had insufficient samples to train the object detector. Looking more closely

at some images in the dataset, shown in Figure 6, it is very difficult to distinguish between certain animals, or between animals and unknown.

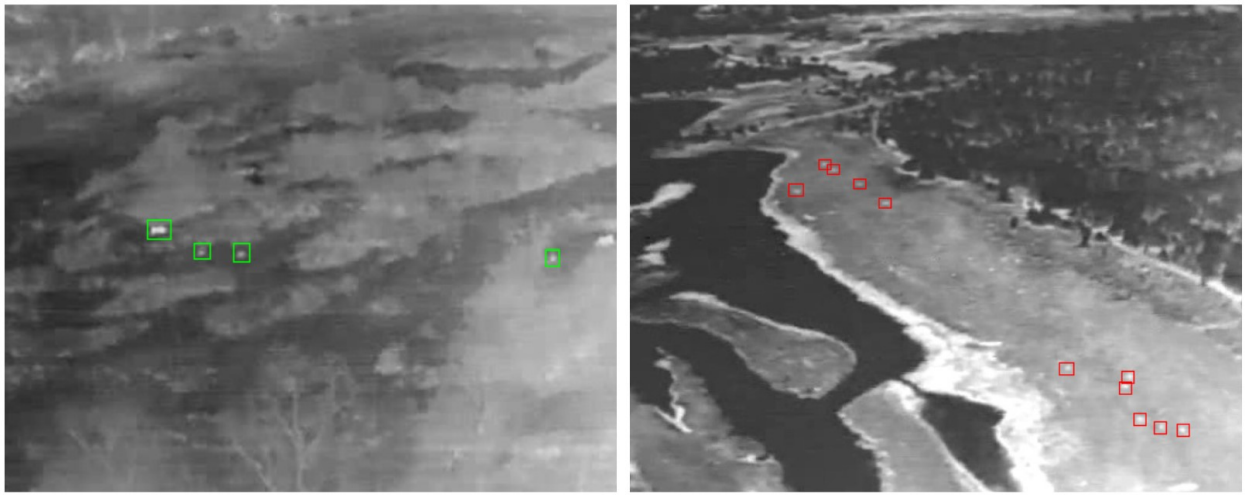

Figure 6: **Difficult to distinguish samples.** Thermal Images showing lions (right) and unknown (left).

As a result of these observations, we conducted several experiments merging different classes. We merged the rare animal classes into unknown, leaving us with three classes – humans, elephants and unknown. More details and the experimental results are given in Section 5. Based on the results from YOLOR trained using these three classes, we decided to move humans into unknown and had 2 classes for our final design prototype. More details will be provided in Section 5.

5 Evaluation and Results

5.1 Implementation Details

All experiments used a batch size of 16 and image size of 640 pixels by 640 pixels. To resize images, we maintained the original aspect ratio and added grey pixel padding to obtain the desired image dimensions. All models were trained for over 100 epochs with an initial learning rate of 0.01, a stochastic gradient descent optimizer with momentum of 0.937 and weight decay of 0.0005. We applied different augmentations to the data to increase diversity, including randomly applying HSV hue, saturation, and value augmentations of 1.5%, 70%, and 40% of the original pixel values, respectively. We also introduced random translations up to 10% of the image dimensions, and scaled images up to a maximum gain of 1.1 or minimum gain of 0.9. As well, we applied divisions to images, creating image mosaics, after which we applied random horizontal flipping with a probability of 0.5. The IoU threshold used across all experiments was 0.2.

5.2 Results and Discussion

Initial experiments were conducted with the combined synthetic and real data, and ten classes to differentiate. The performance is shown in the first row of Table 3. The training data consisted of purely thermal data, without augmentation and filtering techniques mentioned in Section 4.3.1. As a result, performance was poor, with a $mAP_{0.5}$ and $mAP_{0.5:0.95}$ of 7.0% and 2.2%, respectively. Since the training time for experiments with the combined synthetic and real data averaged over five hours per epoch, as mentioned in Section 3, we chose to proceed with training using only real data. Further, to extract more information from the thermal images, we applied the augmentation and filtering techniques mentioned in Section 4.3.1 for the following sets of experiments.

Table 3: **Accuracy comparison.** Results comparison for different optimization methods and architectures. In the "Training Data" column, S stands for *simulation* and R stands for *real*. mAP is measured in %

Architecture	Classes	K-Means Anchors	Augmented Channels	Training Data	$mAP_{0.5}(Epoch)$	$mAP_{0.5:0.95}$
YOLOR	10	✘	✘	S+R	7.0 (42)	2.2
YOLOR	10	✘	✔	R	12.7 (101)	4.8
YOLOR	3	✘	✔	R	25.0 (100)	9.3
YOLOR	2	✘	✔	R	38.2 (121)	13.4
YOLOR	2	✔	✔	R	36.7 (162)	13.3
YOLOv5	2	✘	✔	R	35.4 (187)	12.7

From Table 1, we see the elephants are most represented in all the datasets and they are larger in size compared to all the other animals. Since small animals are very hard to identify and differentiate as seen in Figure 6, we grouped all the smaller animals into the unknown category. Since we wanted to identify humans near animals (as this might indicate poachers), we kept them as a separate category. From Table 3, we see the mAP increases from 12.7% for the 10 classes to 25.0% for three classes.

Even with the improvements, the $mAP_{0.5}$ is still quite low and far from the objective mentioned in Section 3. Through quantitative comparisons, we found that humans, elephants, and unknown categories achieved $mAP_{0.5}s$ of 10.8%, 61.2%, and 24.2%, respectively. After some qualitative inspection (plotting bounding boxes on images), we found while the model performs well for elephants, it struggled with identifying humans vs. unknown. Both humans and unknown are small in size compared to elephants and they look similar. Since performance in differentiating between humans and unknown is poor, deploying a model like this would mean the classifications are unreliable. So, instead of splitting the human and unknown classes, we combined them to achieve more reliable results. If unknown objects are detected near elephants, park services could point patrollers to those areas for inspection, or review the drone footage themselves.

As expected, after the combination of humans and unknown, as the number of classes decreased from ten to three to two, the $mAP_{0.5}$ increased since there is more of a distinction between classes. The largest jump is seen when the number of classes decreased from three to two, as the $mAP_{0.5}$ increased from 25.0% to 38.2%. This is likely because the model no longer had to differentiate between small animals and humans, only small ani-

mals and big animals. It may also have been because there simply were not enough images for any class but elephants. Specifically, for the two class model, the elephant and unknown categories had $mAP_{0.5}$ of 62.3% and 14.1%, respectively.

From Table 3, for two class object detection, we see the YOLOR model initialized using anchors from K-means clustering had a lower $mAP_{0.5}$ of 36.7% than the one initialized with anchors from the most common aspect ratios and sizes in the dataset, which had $mAP_{0.5}$ of 38.2%. Also notable, the mAP peaks at an even higher epoch for the K-means clustering. K-means clustering evenly divides the training data points into clusters, so we were expecting the anchors from K-means clustering to provide a better representation of the dataset. However, it performs worse. This could be because the data points are distributed differently in the training and testing datasets, meaning the initial anchors were not representative of the dataset. Moreover, outliers are likely more of a factor when using a mean as opposed to a modal approach since they skew the mean. The effect of outliers is quite visible in the top middle cluster in Figure 5. As a result, the anchors do not size the bounding boxes correctly.

Since the YOLOR model trained using two classes performed best out of all the YOLOR models, we trained a YOLOv5 model with the same settings and obtained a $mAP_{0.5}$ of 35.5%. This is lower than the $mAP_{0.5}$ of 38.2% for the YOLOR model.

The computation requirements and speed for each of the YOLOR models trained on ten, three, and two classes along with YOLOv5 trained on two classes is shown in Table 4. We can see that while the computation requirements and speed are similar for the all the YOLOR models, YOLOv5 used 20 times less GFLOPs than YOLOR. Nonetheless, both YOLOv5 and YOLOR achieve a performance that remains well within our constraints.

Table 4: **Computation requirements and speed.** Inference speed for various object detectors. Inference denotes time needed to run pre-processing and detection on a single 640×640 image, NMS refers to the time required for non-maximum suppression, and FPS is the inference speed in Frames Per Second.

Architecture	Classes	GFLOPs	Inference (ms)	NMS (ms)	FPS
YOLOR [7]	2	80.38	10.2	3.4	73.6
YOLOR [7]	3	80.40	10.2	3.4	73.6
YOLOR [7]	10	80.50	10.3	3.3	73.6
YOLOv5 [23]	2	4.2	2.5	3.9	156.3

In addition, YOLOR has the proven and reliable downstream benefits as presented in the YOLOR paper [7]. Since YOLOR has been shown to work well for multi-task learning, we can claim with certainty that it will provide runtime benefits over YOLOv5 in potential future tasks like multi-object tracking. On the other hand, YOLOv5 is not proven to have this benefit and is designed specifically for detection tasks. This would mean feature extraction steps in YOLOv5 may not reliably be used for future tasks or yield further runtime benefits. Therefore, with this in mind and considering that YOLOR yields the best performance, we chose to use YOLOR in our final design.

6 Final Design Prototype Overview

Our final selected model is YOLOR with two classes, no k-means anchors, augmented channels, and trained only on real data. Figure 7 shows the diagram of the overall architecture from the input image to the output image with bounding boxes, classes, and confidence scores. We chose YOLOR as the model architecture because it had consistently better results than YOLOv5. For the final prototype only two classes were defined, elephant and unknown. The image data consisted of three channels, the original TIR data, canny edge detection, and adaptive thresholding. The anchor weights were derived by looking at the mode of the data, using the most common aspect ratios and bounding box sizes from the training data annotations.

Our prototype met and even exceeded our two main objectives. It achieved an $mAP_0.5$ of 38.2%, meeting our objective of 35% (and actually almost meets our ambitious proposal objective of 40%). At 80.4 the GFLOPs were well under the set maximum of 1330, and the FPS of 73.6 significantly exceeded our goal of 10 FPS.

Mitigation Strategy Reflection. From the success of our design prototype, it is evident that our mitigation strategy of merging classes worked well. The reason we needed to employ this strategy was limited compute, which led to training on only the real data, which in turn led to class imbalance.

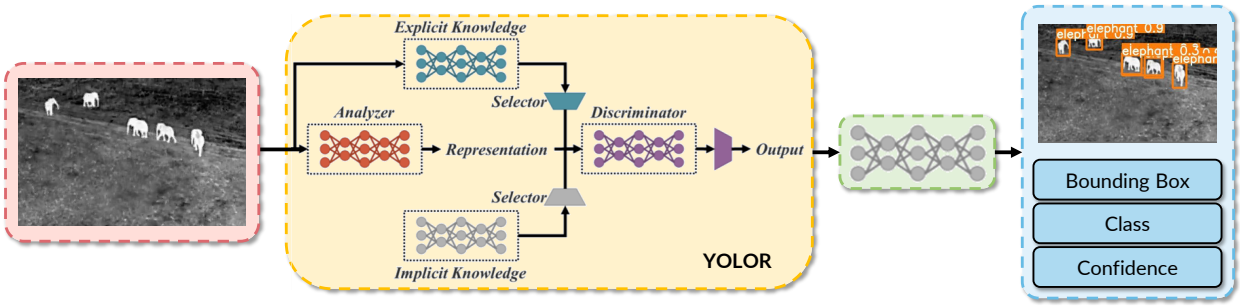

Figure 7: **Design Architecture**. YOLOR (architecture adapted from [7]) presents a unified network where one feature representation is learned for accomplishing multiple tasks. We refer readers to [7] for further details.

6.1 Precision-Recall Analysis

In analysing the plot in Figure 8, we see that with recall between 0.3 and 0.6, increasing recall will not drastically affect precision. However, increasing or decreasing recall values outside of the aforementioned range will cause drastic changes in precision. A high precision means a low false positive rate (no incorrect detections), while high recall means a low false negative rate (no missed detections). For the elephant P/R curve (upper grey curve), we can see that there is a large area under the curve ratio. This corresponds to a high precision and high recall, which means that elephants can be detected accurately without missing detections. However, for the unknown category (lower grey curve), the

area under the curve is small, which means that unknown objects are often missed, or spurious objects are detected as unknown.

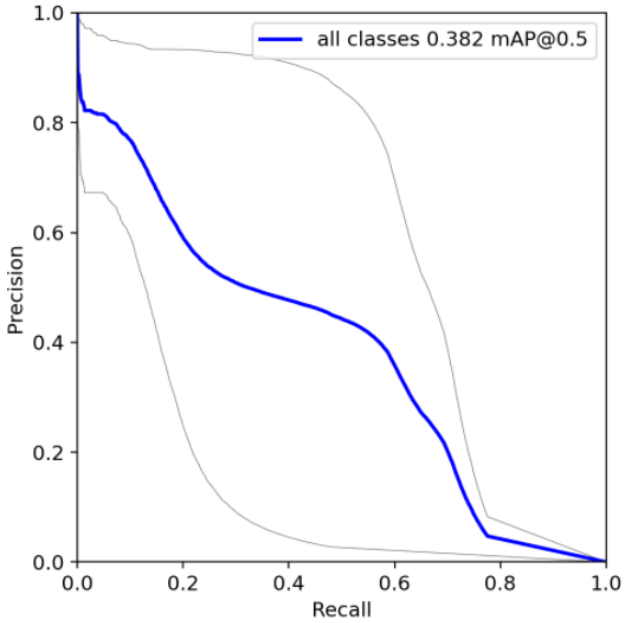

Figure 8: **Performance assessment.** Precision and recall curve of YOLOR. Highlighted blue curve denotes PR for all classes, top grey curve denotes PR for the elephant class, and the bottom grey curve denotes PR for the unknown class.

6.2 Failures and Success Cases

To qualitatively assess the results of the final prototype, we have identified cases where the model performs successfully (Figure 9) and cases where it fails (Figure 10).

From cases with successful detection, we can observe the model correctly identifies objects in well contrasted areas or in isolation, regardless of their size. Furthermore, we see that the model actually identifies animals that seem to have been missed when the dataset labeling was done, indicating that the model performs better than human labeling. This can be clearly seen in the third row of images in Figure 9.

Even if the model performs well for some cases, there are failure modes. From the first row of images in Figure 10, we see that motion blur causes deformations to the animal shapes, which results duplicate or missing bounding boxes. Motion blur is a pretty big concern since it results in images that show "ghost" animals. Since the model has learned to identify ghost animals (an example is shown in the first row of Figure 9), it identifies these hypothetical animals created by motion blur as real animals.

As seen in the second row of Figure 10, another failure mode comes from the model trying to identify objects that are very small or blend in with the background. From Figure 6, we saw that the training dataset contains small animals that appear as small and only slightly contrasting circles. In the third row of Figure 10, we see spurious detection in

areas of roughness and low contrast (unknown object identified on the bottom of the image). The model likely sees it as similar to the small slightly bright circle it has learned to identify as unknown during training.

From the fourth row of Figure 10, we see the model fails to identify a human. It does correctly identify the large elephant but has difficulty with the smaller human. Lastly, from the fifth row in Figure 10, the model fails to clearly identify animals in images with occlusion and animals that overlap in herds. It identifies them as multiple cluttered detections.

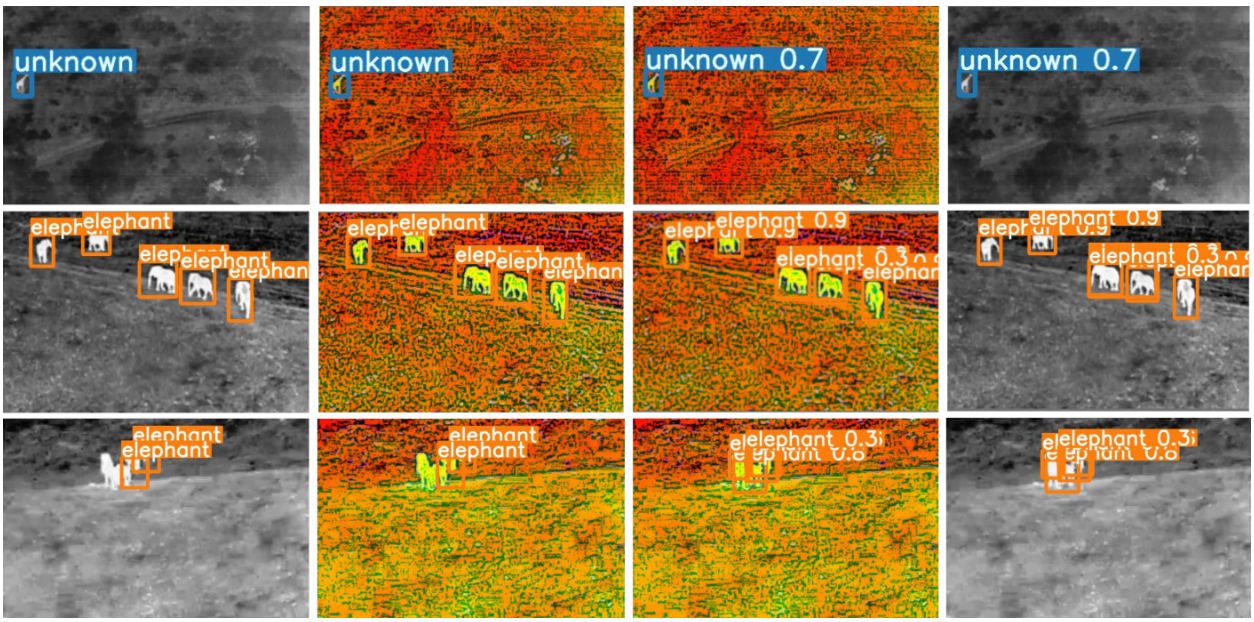

Figure 9: **Success Cases.** Success cases for final prototype, note that this is only a small sample included for demonstration purposes. From left to right, initial thermal image with ground truth labels, pre-processed image with added augmented channels and ground truth labels, augmented channels with predicted labels, ground truth image with predicted labels. Note that inference was run using augmented channels (center pair), the black and white thermal images are included as references.

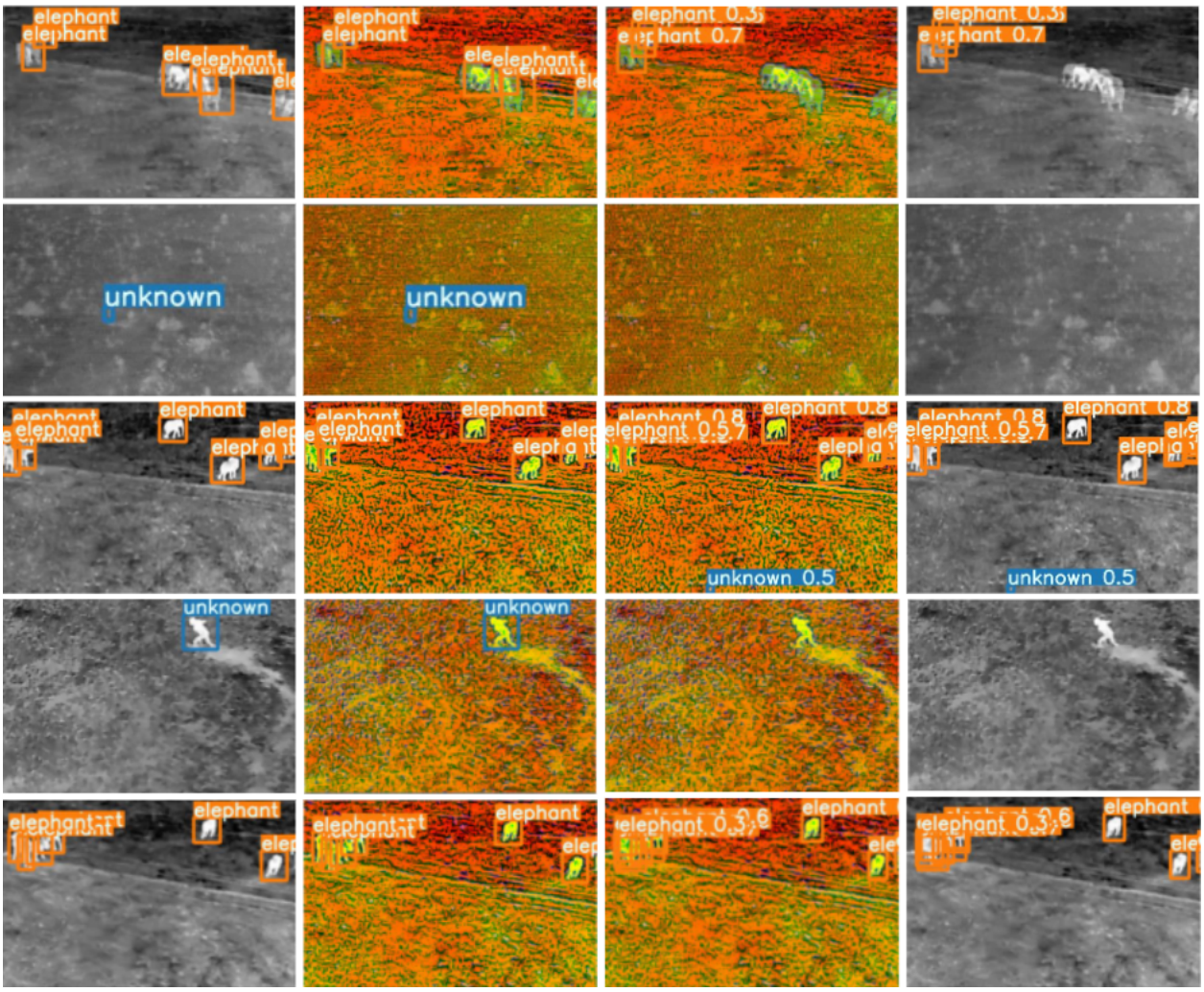

Figure 10: **Failure cases.** Different types of failure modes for final prototype. From left to right, initial thermal image with ground truth labels, pre-processed image with added augmented channels and ground truth labels, augmented channels with predicted labels, ground truth image with predicted labels. Note that inference was run using augmented channels (center pair), the black and white thermal images are included as references.

6.3 Future Extensions

To scale this prototype to a real-world implementation, we consider the hardware we assumed we would have the Nvidia TX2 3.2.3. Since this processing system includes a GPU, we can directly use our trained model onboard the UAV. To further optimize resource usage, a next step could be to explore model compression techniques [61] like parameter pruning, quantization, and knowledge distillation to decrease model size and memory usage. This would also allow models for other tasks to be run on the GPU at the same time. Another extension would be to add multi-object tracking, which would allow us to track the movement of herds as well as gain other useful insights.

7 Lessons Learned

There were a number of experiences over the course of completely this project which taught us valuable lessons.

The first challenge was attempting to chose a topic, which required careful balancing and compromise among the group members' interests. Our agreed upon common interests ended up being UAVs and environmentalism, which led us to wildlife protection using UAVs. We also spent a significant amount of time discussing what might be suitable, and how difficult or innovative the project needed to be. Choosing our own task may be reflective of certain real-world scenarios (for example, coming up with a start-up idea), so this project has prepared us for that. If we were to repeat this course, we would pick a topic before the semester started. This would allow us to maximize the time we have to develop our design, leading to an even better final prototype. This extra time might have allowed as to test a third object detection architecture, CenterNet2.

One lesson learned is that data (and thus the dataset choice) is critical to the validation and verification process. In particular, we experienced first-hand the impact that class imbalance has on experimental results. This relates to the importance of simulation data for class balancing and data diversity when there is a lack of real data. Now that we understand the gravity of class imbalance and need for simulation data, we would invest more time towards examining and dealing with the data if we were to attempt this project again.

Another lesson we learned was that thermal images are very difficult to work with. They only have one channel, unlike RBG images which have three for each colour and thus the object detector has a third of the information. Also, animals tend to give off similar levels of heat, translating to similar pixel intensities, meaning the detector focused on the shape. In contrast, with an RGB image, colour could also be used to differentiate between animals. Furthermore, aerial images tend to be blurry because the drone is moving, and animals can appear very small because they are very far away. From aerial viewpoints farther away, shapes of different animals can begin to look the same. This would make it difficult for the detector to differentiate between animal classes, and it is often impossible for the human eye to make a distinction.

Furthermore, a lesson that we learned was the importance of reliable compute for a deep learning-based project. One of the most difficult challenges throughout the experimentation stage was limited compute resources. We have mentioned this at greater length throughout the report, but it did prevent us from being able to train with the synthetic data, reducing the number of samples for each animal and thus limiting the prototype's ability to generalize. It also meant we had to reduce the number of classes because for certain classes there were no or very few examples in the real data. In the future, selecting a scope that is compatible with the available compute resources would benefit the overall result of any design.

References

[1] National Geographic, "The Global Impacts of Habitat Destruction," https://blog.nationalgeographic.org/2019/09/25/the-global-impacts-of-habitat-destruction/.

[2] L. E. Coristine and J. T. Kerr, "Habitat loss, climate change, and emerging conservation challenges in canada," *Canadian Journal of Zoology*, vol. 89, no. 5, pp. 435–451, 2011. [Online]. Available: https://doi.org/10.1139/z11-023

[3] C. Zoe, "The technology fighting poachers." [Online]. Available: https://www.bbcearth.com/news/the-technology-fighting-poachers

[4] World Animal Protection, "Canada's role in the wildlife trade," https://www.worldanimalprotection.ca/news/canadas-role-wildlife-trade, Aug 2020.

[5] Elise von Scheel, "Black market animal smuggling is booming in Canada," https://www.cbc.ca/news/politics/black-market-animal-smuggling-canada-1.4720102, June 2018.

[6] "Conservation drones." [Online]. Available: https://conservationdrones.org/

[7] C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao, "You only learn one representation: Unified network for multiple tasks," 2021. [Online]. Available: https://arxiv.org/abs/2105.04206

[8] E. Bondi, R. Jain, P. Aggrawal, S. Anand, R. Hannaford, A. Kapoor, J. Piavis, S. Shah, L. Joppa, B. Dilkina, and M. Tambe, "BIRDSAI: A Dataset for Detection and Tracking in Aerial Thermal Infrared Videos," in *WACV*, 2020.

[9] "What's The Difference between Thermal Imaging and Night Vision?" https://www.flir.ca/discover/ots/thermal-vs-night-vision/.

[10] "What's The Difference between Thermal Imaging and Night Vision?" https://www.agmglobalvision.com/Night-Vision-vs-Thermal-Optics, Mar 2020.

[11] A. Jo, J.-S. Park, Y.-H. Seo, and G.-J. Jang, "Performance improvement of human detection in thermal images using principal component analysis and blob clustering," *The Journal of the Institute of Webcasting, Internet and Telecommunication*, vol. 13, 04 2013.

[12] E. Bernard, N. Rivière, M. Renaudat, M. Péalat, and E. Zenou, "Active and thermal imaging performance under bad weather conditions," in *6th International Symposium on Optronics in Defence and Security*, 2014.

[13] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *Conference on Computer Vision and Pattern Recognition*, 2001. [Online]. Available: https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf

[14] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Conference on Computer Vision and Pattern Recognition*, 2004. [Online]. Available: https://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf

[15] M. Z. Rodriguez, C. H. Comin, D. Casanova, O. M. Bruno, D. R. Amancio, L. d. F. Costa, and F. A. Rodrigues, "Clustering algorithms: A comparative approach," *PLOS ONE*, vol. 14, no. 1, pp. 1–34, 01 2019. [Online]. Available: https://doi.org/10.1371/journal.pone.0210236

[16] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," 2014.

[17] R. Girshick, "Fast r-cnn," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1440–1448.

[18] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," 2015. [Online]. Available: https://arxiv.org/abs/1506.01497

[19] Y. Zhong, J. Wang, J. Peng, and L. Zhang, "Anchor box optimization for object detection," 2018. [Online]. Available: https://arxiv.org/abs/1812.00469

[20] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," 2016. [Online]. Available: https://arxiv.org/abs/1612.08242

[21] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," 2018.

[22] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[23] G. Jocher, A. Chaurasia, A. Stoken, J. Borovec, NanoCode012, Y. Kwon, TaoXie, J. Fang, imyhxy, K. Michael, Lorna, A. V, D. Montes, J. Nadar, Laughing, tkianai, yxNONG, P. Skalski, Z. Wang, A. Hogan, C. Fati, L. Mammana, AlexWang1900, D. Patel, D. Yiwei, F. You, J. Hajek, L. Diaconu, and M. T. Minh, "ultralytics/yolov5: v6.1 - TensorRT, TensorFlow Edge TPU and OpenVINO Export and Inference," Feb. 2022. [Online]. Available: https://doi.org/10.5281/zenodo.6222936

[24] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as points," *CoRR*, vol. abs/1904.07850, 2019. [Online]. Available: http://arxiv.org/abs/1904.07850

[25] A. N and U. S V, "Drone object detection using deep learning algorithms," in *2021 Third International Conference on Inventive Research in Computing Applications (ICIRCA)*, 2021, pp. 1187–1192.

[26] J. Lee, J. Wang, D. Crandall, S. Šabanović, and G. Fox, "Real-time, cloud-based object detection for unmanned aerial vehicles," in *2017 First IEEE International Conference on Robotic Computing (IRC)*, 2017, pp. 36–43.

[27] R. Walambe, A. Marathe, and K. Kotecha, "Multiscale object detection from drone imagery using ensemble transfer learning," *Drones*, vol. 5, no. 3, 2021. [Online]. Available: https://www.mdpi.com/2504-446X/5/3/66

[28] W. Andrew, C. Greatwood, and T. Burghardt, "Visual localisation and individual identification of holstein friesian cattle via deep learning," in *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, 2017, pp. 2850–2859.

[29] A. Rivas, P. Chamoso, A. González-Briones, and J. M. Corchado, "Detection of cattle using drones and convolutional neural networks," *Sensors*, vol. 18, no. 7, 2018. [Online]. Available: https://www.mdpi.com/1424-8220/18/7/2048

[30] N. Rey, M. Volpi, S. Joost, and D. Tuia, "Detecting animals in african savanna with uavs and the crowds," *Remote Sensing of Environment*, vol. 200, pp. 341–351, 2017.

[31] E. Okafor, R. Smit, L. Schomaker, and M. Wiering, "Operational data augmentation in classifying single aerial images of animals," in *2017 IEEE International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*, 2017, pp. 354–360.

[32] J. C. van Gemert, C. R. Verschoor, P. Mettes, K. Epema, L. P. Koh, and S. Wich, "Nature conservation drones for automatic localization and counting of animals," in *Computer Vision - ECCV 2014 Workshops*, L. Agapito, M. M. Bronstein, and C. Rother, Eds. Springer International Publishing, 2015, pp. 255–270.

[33] C. Chalmers, P. Fergus, C. A. Curbelo Montanez, S. N. Longmore, and S. A. Wich, "Video analysis for the detection of animals using convolutional neural networks and consumer-grade drones," *Journal of Unmanned Vehicle Systems*, vol. 9, no. 2, pp. 112–127, 2021. [Online]. Available: https://doi.org/10.1139/juvs-2020-0018

[34] B. Kellenberger, D. Marcos, and D. Tuia, "Detecting mammals in uav images: Best practices to address a substantially imbalanced dataset with deep learning," *arXiv:1806.11368*, 01 2018.

[35] S.-J. Hong, Y. Han, S.-Y. Kim, A.-Y. Lee, and G. Kim, "Application of deep-learning methods to bird detection using unmanned aerial vehicle imagery," *Sensors*, vol. 19, no. 7, 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/7/1651

[36] E. Corcoran, S. Denman, J. Hanger, B. Wilson, and G. Hamilton, "Automated detection of koalas using low-level aerial surveillance and machine learning," *Scientific Reports*, vol. 9, 2019. [Online]. Available: https://doi.org/10.1038/s41598-019-39917-5

[37] C. Burke, M. Rashman, S. Longmore, O. McAree, P. Glover-Kapfer, M. Ancrenaz, and S. Wich, "Successful observation of orangutans in the wild with thermal-equipped drones," *Journal of Unmanned Vehicle Systems*, vol. 7, 05 2019.

[38] C. Burke, M. Rashman, S. Wich, A. Symons, C. Theron, and S. Longmore, "Optimizing observing strategies for monitoring animals using drone-mounted thermal infrared cameras," *International Journal of Remote Sensing*, vol. 40, no. 2, pp. 439–467, 2019. [Online]. Available: https://doi.org/10.1080/01431161.2018.1558372

[39] E. Bondi, F. Fang, M. Hamilton, D. Kar, D. Dmello, J. Choi, R. Hannaford, A. Iyer, L. Joppa, M. Tambe, and R. Nevatia, "Spot poachers in action: Augmenting conservation drones with automatic detection in near real time," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Apr 2018. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/11414

[40] R. Padilla, S. Netto, and E. da Silva, "A survey on performance metrics for object-detection algorithms," in *2020 International Conference on Systems, Signals and Image Processing*, 07 2020.

[41] Skydio, "Skydio X2E Color/Thermal Datasheet," https://pages.skydio.com/rs/784-TUF-591/images/skydio-x2e-datasheet.pdf.

[42] senseFly, "eBee Geo," https://www.sensefly.com/app/uploads/2021/02/eBee-GEO-EN-2021-V1.pdf.

[43] NVIDIA, "Jetson TX2 Module," https://developer.nvidia.com/embedded/jetson-tx2.

[44] Golden Gate Audubon Society, "Drone Dangers and Birds," https://goldengateaudubon.org/conservation/make-the-city-safe-for-wildlife/drone-dangers-and-birds/.

[45] Amy Graff, "'One-in-a-million incident': The mystery of a dead hawk at Dolores Park," https://www.sfgate.com/bayarea/article/Drone-killed-hawk-Dolores-Park-San-Francisco-14962557.php.

[46] D. Du, Y. Qi, H. Yu, Y. Yang, K. Duan, G. Li, W. Zhang, Q. Huang, and Q. Tian, "The unmanned aerial vehicle benchmark: Object detection and tracking." in *ECCV*, 2018.

[47] S. Li and D.-Y. Yeung, "Visual object tracking for unmanned aerial vehicles: A benchmark and new motion models," in *AAAI*, 2017.

[48] G.-S. Xia, X. Bai, J. Ding, Z. Zhu, S. Belongie, J. Luo, M. Datcu, M. Pelillo, and L. Zhang, "Dota: A large-scale dataset for object detection in aerial images," in *CVPR*, 2018.

[49] E. Bondi, D. Dey, A. Kapoor, J. Piavis, S. Shah, F. Fang, B. Dilkina, R. Hannaford, A. Iyer, L. Joppa, and M. Tambe, "Airsim-w: A simulation environment for wildlife conservation with uavs," in *ACM SIGCAS Conference on Computing and Sustainable Societies*, 2018.

[50] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *ECCV*, 2014.

[51] X. Zhou, V. Koltun, and P. Krähenbühl, "Probabilistic two-stage detection," 2021. [Online]. Available: https://arxiv.org/abs/2103.07461

[52] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 02, pp. 318–327, feb 2020.

[53] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," in *CVPR*, 2020.

[54] G. Yu, Q. Chang, W. Lv, C. Xu, C. Cui, W. Ji, Q. Dang, K. Deng, G. Wang, Y. Du, B. Lai, Q. Liu, X. Hu, D. Yu, and Y. Ma, "Pp-picodet: A better real-time object detector on mobile devices," 2021. [Online]. Available: https://arxiv.org/abs/2111.00902

[55] C.-Y. Wang, H.-Y. M. Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh, "Cspnet: A new backbone that can enhance learning capability of cnn." in *CVPRW*, 2020.

[56] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *CVPR*, 2018.

[57] U. Nepal and H. Eslamiat, "Comparing yolov3, yolov4 and yolov5 for autonomous landing spot detection in faulty uavs," *Sensors*, vol. 22, no. 2, p. 464, 2022.

[58] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1986.

[59] OpenCV, "Open source computer vision library," https://github.com/opencv/opencv, 2022.

[60] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, 1979.

[61] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "A survey of model compression and acceleration for deep neural networks," *arXiv:1710.09282*, 2017.