LIVERPOOL JOHN MOORES UNIVERSITY

LJMU Research Online

Chalmers, C, Fergus, P, Curbelo Montañez, CA, Longmore, SN and Wich, SA

Video Analysis for the Detection of Animals Using Convolutional Neural Networks and Consumer-Grade Drones

http://researchonline.ljmu.ac.uk/id/eprint/14813/

Article

Citation (please note it is advisable to refer to the publisher's version if you intend to cite from this work)

Chalmers, C, Fergus, P, Curbelo Montañez, CA, Longmore, SN and Wich, SA (2021) Video Analysis for the Detection of Animals Using Convolutional Neural Networks and Consumer-Grade Drones. Journal of Unmanned Vehicle Systems. ISSN 2291-3467

LJMU has developed **LJMU Research Online** for users to access the research output of the University more effectively. Copyright © and Moral Rights for the papers on this site are retained by the individual authors and/or other copyright owners. Users may download and/or print one copy of any article(s) in LJMU Research Online to facilitate their private study or for non-commercial research. You may not engage in further distribution of the material or use it for any profit-making activities or any commercial gain.

The version presented here may differ from the published version or from the version of the record. Please see the repository URL above for details on accessing the published version and note that access may require a subscription.

For more information please contact researchonline@ljmu.ac.uk

http://researchonline.ljmu.ac.uk/

Video Analysis for the Detection of Animals Using Convolutional Neural Networks and Consumer-Grade Drones

C. Chalmers[1], P.Fergus[1], C. Aday Curbelo Montanez[1], Steven N. Longmore[2], and Serge A. Wich[3]

[1]School of Computer Science, Liverpool John Moores University, Liverpool, UK.

[2]Astrophysics Research Institute, Liverpool John Moores University, Liverpool, UK

[3]School of Biological and Environmental Sciences, Liverpool John Moores University, Liverpool, UK

ABSTRACT

Determining animal distribution and density is important in conservation. The process is both time-consuming and labour-intensive. Drones have been used to help mitigate human-intensive tasks by covering large geographical areas over a much shorter timescale. In this paper we investigate this idea further using a proof of concept to detect rhinos and cars from drone footage. The proof of concept utilises off-the-shelf technology and consumer grade drone hardware. The study demonstrates the feasibility of using machine learning (ML) to automate routine conservation tasks such as animal detection and tracking. The prototype has been developed using a DJI Mavic Pro 2 and tested over a Global System for Mobile Communications (GSM) network. The Faster RCNN Resnet 101 architecture is used for transfer learning. Inference is performed with a frame sampling technique to address the required trade-off between precision, processing speed and live video feed synchronisation. Inference models are hosted on a web platform and video streams from the drone (using OcuSync) are transmitted to a Real-Time Messaging Protocol (RTMP) server for subsequent classification. During training, the best model achieves a Mean Average Precision (mAP) of 0.83, Intersection Over Union @(IOU) 0.50 and 0.69 @IOU 0.75, respectively. On testing the system in Knowsley Safari our prototype was able to achieve the following: Sensitivity (Sen): 0.91(0.869,094), Specificity (Spec): 0.78(0.74,0.82) and an Accuracy (ACC): 0.84 (0.81,0.87) when detecting rhinos, and Sen: 1.00(1.00,1.00), Spec: 1.00(1.00,1.00) and an ACC: 1.00(1.00,1.00) when detecting cars.

Key words— Conservation, Deep Learning, Convolutional Neural Networks, Inferencing, Drone Technology

INTRODUCTION

Biodiversity is under threat and this is caused by a multitude of factors (Maxwell et al. 2016). To facilitate and evaluate conservation management, data on animal distributions and density are crucial (Nichols and Williams 2006). However, traditional ground-based methods such as line transects are costly to conduct over large areas and the problem is further exacerbated as tasks need to be repeated frequently to be able to determine trends over time (Buckland et al. 2004, 2001). Alternative methods such as camera traps or passive acoustic monitoring are promising but are also expensive to implement over large areas (Crunchant et al. 2020). Recently, conservation scientists have begun using drones to facilitate data collection on animal distributions and density (Chabot and Bird 2015; Christie et al. 2016; Wich and Koh 2018). Although drones can rapidly obtain large amounts of data, processing these data is human-centric (see table 3.3 in Wich and Koh (2018)) leading to high-cost, greater time investments and difficulties in near real-time detections. As a result, there have been efforts to automate the detection of animals, humans, cars, and other objects such as nests (Longmore et al. 2017; Bondi et al. 2019; Bondi et al. 2018; Fang et al. 2016; Maire, Alvarez, and Hodgson 2015; van Gemert et al. 2014; Lamba et al. 2019; Peng et al. 2020).

Detecting objects of interest has been made easier by utilising low-cost consumer drones, computer vision, Deep Learning (DL) and High-Performance Computing (HPC). Examples of computer vision applications that have already been successfully used include those in medicine (Saria, Butte, and Sheikh 2018), manufacturing (Martinez, Ahmad, and Al-Hussein 2019), and engineering (Jakobs, Weber, and Stapp 2019). We can capitalise on the advances made in these fields and utilise them to support conservation tasks. DL has facilitated developments in image processing, none more so than the availability of frameworks like TensorFlow (Rampasek and Goldenberg 2016), Microsoft Cognitive Tool Kit (CNTK) (Banerjee, Hamidouche, and Panda 2016) and Caffe 2 (Hazelwood et al. 2018). Convolutional Neural Networks (CNNs), which are DL algorithms, have become the gold standard in image processing tasks that are focused on object detection and segmentation. These types of networks are considered important tools for detecting and analysing animals in video footage (LeCun et al. 1999). The computational requirements to train and host complex CNNs, such as the Faster-Region-based Convolutional Neural Network (Faster-RCNN), is a significant challenge in many applications. This is particularly true when real-time video processing is required, such as in conservation settings. These challenges have been alleviated through advancements in Graphical Processing Units (GPUs). GPUs are now heavily utilised by the computer vision community, particularly by those interested in deep learning and reducing model training and inference time (Lim et al. 2017).

Given the computational requirements, many current approaches in conservation have utilised machine learning to process images after the data collection has been completed (Lamba et al. 2019; Bondi et al. 2019; Wich and Koh 2018; Bondi et al. 2018; Longmore et al. 2017; Fang et al. 2016; Maire, Alvarez, and Hodgson 2015; van Gemert et al. 2014). However, there are instances when near real-time detections are required, such as in animal surveys, animal tracking, and asset management. In these situations, Machine Learning (ML) needs to be applied as and when images are being collected. This can be challenging to achieve depending on the field conditions where communication and hardware could be limited. Where GSM networks are available, images can be streamed live from the drone to a server to conduct the analyses (online inference). However, many environments do not have GSM availability, and as such, off-line inferencing can be conducted by transmitting real-time data from the drone to a laptop in the field. In a similar way to (Bondi et al. 2018) we explore the first approach, but additionally we also discuss the usage of a laptop in the field to conduct the machine learning. Our study differs from the (Bondi et al. 2018) approach in that we use a consumer-grade DJI Mavic 2 drone unlike their custom build solution. This makes our approach a cheaper and more scalable solution for animal habitat screening. This is important in dangerous and difficult to reach habitats, for example when tracking poachers in hostile environments. In

this paper we demonstrate that using this approach, near real-time detections of objects using machine learning was achieved. This may be a useful tool within conservation settings (animal surveys, poaching, logging, etc.,) where near-real time detection is required. It is hoped that this will simultaneously provide a first step approach for further research and development, while also benefitting both conservation managers and scientists.

METHODOLOGY

This section describes the dataset adopted in this study and the steps taken to: a) pre-process the data, b) train the model, and c) evaluate its performance. The pipeline presented is used in three experiments to determine the applicability of the proposed system. The methodology is designed around three research questions a) can specific classes be detected in video footage, b) can number of distinct classes within the video footage be counted and c) can detections be performed in real-time.

The first experiment evaluates the trained model using video footage of rhinos obtained from YouTube and determines how well the model generalises across diverse environments. The results are then used to fine-tune the model before it is deployed into the live environment. While this is less than ideal, it is an important step given that access to animals in real-world environments (particularly rare and endangered animals) is difficult and in many cases not possible due to regulations and restrictions. We show that pre-recorded footage of animals in their natural habitat is a sufficient alternative to help evaluate and train deep learning models. It is important to note however that this approach may only be efficient in cases where video footage is available for the desired species. For example, it may not be possible to obtain video footage of rare and under reported animals in their natural habitat.

The second experiment is designed to evaluate the model's ability to generalise on unseen data and process the video footage obtained from a drone in real-time. The third experiment was conducted at Knowsley Safari in the UK[1]. It is important to note that the safari environment does not fully represent the animals' natural environment. Nonetheless, safari parks do present a viable alternative for initial testing. Adopting this approach is important for the reasons discussed above – it is not always possible to carry out evaluations in real habitats due to regulations and restrictions. In this experiment a DJI Mavic Pro 2 is used to stream video footage from the rhino enclosure at an altitude of 60 meters. Images are processed using the ML pipeline and then evaluated using the performance metrics outlined in this section. The two main features that will be implemented will be image classification and the second will focus on induvial object counting.

Data capture

A single data set was constructed and used in the experiments reported in this study. This consisted of 700 images. The dataset comprised both RGB and thermal (colour and grey scale palettes) images. The data contained two classes: rhinos and cars. Each class had 350 images with resolutions between 300 x 147 and 3840 x 2160 pixels (which were scaled automatically to 1500 x 1500 pixels). Noise was introduced into the training process by combining aerial footage with close exposure and thermal images as shown in Figure 1. The aerial RGB images were captured using a drone, while both the thermal and grey scale images were captured using a ground-based camera. To supplement our own field trial data, RGB images of a resolution of 1024 x 768 were downloaded from Google and combined. In total 175 images per class were obtained from drone-based data, 85 images per class were obtained from ground-based cameras and the remaining 90 images per class were downloaded from google as shown in Table 1. This is an important step in DL that maximises the complexity in the learning process. Using a broad mixture of images from aerial and ground-based representations of objects within different environmental settings forces the network to learn the most important features within the objects of interest. Thermal and grey scale images were captured using a FLIR

[1] https://www.knowsleysafariexperience.co.uk/

One smart phone thermal camera. The resolution for both the thermal and grey scale images was set to 640 x 512 pixels. These images were collected at Knowsley Safari. The data were subsequently split into three separate datasets; one for training, one for validation, and one for testing. It is important to note that the testing dataset is not used during the training of the model.

Tagging of the data was undertaken using the Visual Object Tagging Tool (VoTT) version 1.7.0[2]. Bounding boxes were used to identify regions of interest. All tagged regions within each image were exported as Extensible Mark-up Language (XML) in TensorFlow Pascal VOC format. The generated XML was converted into Comma Separated Values (CSV) using the Python libraries Pandas and XML to parse the XML data. Using TensorFlow 1.13.1 and Pillow, both the CSV and image data were converted into the TFRecord format ready for training.

Model selection

The Faster-RCNN network architecture was implemented to perform object detection in two distinct stages (Talukdar et al. 2018). The first stage used Region Proposal Networks (RPNs) to identify and extract features from the selected layers. This allows the model to estimate bounding box locations. The second stage conducted further processing to adjust the localisation of the bounding box by minimising the selected loss function. Both the region proposal and object detection tasks were undertaken by the same CNN. Figure 2 shows the basic architecture of a Faster-RCNN. This model architecture allows us to perform image classification and object detection for counting. This architecture has the added benefit of supporting a wide range of object including those that are occluded and where the same objects are often grouped together which is useful for object counting.

Transfer learning

Through empirical analysis we found that a dataset comprising 350 images per class and transfer learning produced sufficient results to demonstrate the applicability of the solution presented in this paper. Transfer learning significantly reduces the amount of data required to train a model while maintaining a sufficient level of accuracy. This is an important aspect of this work that enables the development of such solutions to become attainable to a broad range of users with limited data, hardware, and financial support.

Transfer learning enables us to adopt a pre-trained model and fine-tune its learned parameters to support new objects of interest. This allows the new images in the dataset to be learned and used in inferencing tasks (Shin et al. 2016). This is an important technique as training CNNs on small datasets leads to poor performance due to low variance. Therefore, building on a much larger robust model that has been trained using a large number of images, helps to counteract the issues associated with the low sample size used in this study. The base model adopted for the transfer learning tasks was the Faster-RCNN Resnet 101 model which has been pre-trained on the COCO dataset. COCO is a large object detection dataset containing 330 thousand images and 1.5 million object instances[3].

Model training

Model training was conducted on a HP ProLiant ML 350 Gen 9 server. The server had x2 Intel Xeon E5-2640 v4 series processors and 768GB of RAM. An additional GPU stack comprising x4 NVidia Quadro M4000 graphics cards with a combined total of 32GB of DDR5 RAM was installed. TensorFlow 1.13.1, CUDA 10.0 and CuDNN version 7.5 formed the software aspects of the training pipeline. In the pipeline.config file used by TensorFlow the following training parameters were set:

[2] https://github.com/microsoft/VoTT
[3] http://cocodataset.org/#home

- The aspect ratio resizer minimum and maximum coefficients were set to 1500 x 1500 pixels, respectively. This minimised the scaling effect on the acquired data. Increasing the resolution further will facilitate greater accuracy but would impact the computational limitations of the training platform.
- The default for the feature extractor coefficient was retained to provide a standard 16-pixel stride length to maintain a high-resolution aspect ratio.
- The batch size coefficient was set to one to maintain GPU memory limits.

The Resnet 101 model implements the Adam optimiser to minimise the loss function (Kingma and Ba 2014). Unlike other optimisers such as Stochastic Gradient Decent (SGD), which maintains a single learning rate (alpha) throughout the entire training session, Adam calculates the moving average of the gradient m_t/squared gradients v_t and the parameters beta1/beta2 to dynamically adjust the learning rate. Adam, as described in (Heusel et al. 2017), is defined as:

$$m_t = \beta_1 m_t - 1 + (1 - \beta_1) g_t \quad (1)$$
$$v_t = \beta_2 v_t - 1 + (1 - \beta_2) g_t^2$$

Where m_t and v_t are the estimates of the first and second moment of the gradients. Both m_t and v_t are initialised with 0's. Biases are corrected by computing the first and second moment estimates (Heusel et al. 2017):

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \quad (2)$$
$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

Parameters are updated using the Adam update rule:

$$\theta_{t+1} = \theta_t - \frac{n}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t. \quad (3)$$

The ReLU activation function was adopted during training to provide improvements over other functions such as sigmoid or hyperbolic tangent (tanh) activations that suffer from saturation changes around the mid-point of their input which reduces the amount of available tuning. The use of these in deep multi-layered networks result in ineffective training caused by a vanishing gradient (Kanai, Fujiwara, and Iwamura 2017). ReLU, as defined in (Ba and Frey 2013), is:

$$g(x) = max(0, x) \quad (4)$$

Inferencing pipeline

The object detection system proposed in this study interfaces with a variety of camera systems using the Real-Time Messaging Protocol (RTMP). The Mavic Pro 2 drone system used in the experiments is capable of transmitting 4K videos at 30fps over a distance in excess of 7 kilometres (km) to a linked controller using the OcuSync 2.0 protocol. Video streams are re-directed from the controller using a local Wi-Fi connection to a field laptop or to a remote server using GSM. Object detection on video frames is performed on the laptop or remote server. Figure 3 illustrates the end-to-end inferencing pipeline.

The video stream from the controller is wirelessly transmitted to an RTMP server hosted by NGNIX. Each frame is processed in Python using OpenCV and serialised with a flask web server. Each video frame is inferenced with a TensorFlow backend model using a loaded frozen inference graph. The bounding box detections are returned to the flask object detection site where the operator monitors the results. Where local

processing is used (field laptop), the complete framework is run locally on the laptop. In this study, local inferencing is performed using a Dell XPS laptop with an Intel i9 CPU and Nvidia 1050Ti GPU, while online inferencing is undertaken using a custom-built server containing an Intel Xeon E5-1630v3 CPU, 64GB of RAM and a NVidia Quadro RTX 8000 GPU. TensorFlow 1.13.1, CUDA 10 and CuDNN 7.5 represented the software components used for inferencing. All the required services and components are available through a public facing conservation AI portal[4] developed by the research team.

The Faster-RCNN provided the necessary capabilities to address the two main research question in this paper which are a) image classification in video footage and b) unique class counting. However, in answering the third research question there is a need to overcome the inferencing limitations associated with Faster-RCNNs, and the requirement to provide real-time object detection, a frame sampling technique is proposed. A single adjustable parameter specifies the number of frames to be inferenced within a given time period. This results in frame skipping which maintains video speed integrity and accurate bounding box overlay, i.e. real-time synchronisation with live video streams. This is a configurable parameter that can be adjusted within the object detection site to meet the needs of the underlying GPU architecture. The frame sampling technique is outlined in algorithm 1.

Algorithm 1 Dynamic Frame Sampling

Data: Frame Read Correctly *ret*,
Current Video Frame *frame*,
Inferencing Parameter *fpsLimitStream*,
Start Time *startTime*,
Now Time *nowTime*
Result: Tracked Bounding Boxes *inference* in *frame*
Read *ret, frame*
Set *startTime*
While *ret* = TRUE
Set *nowTime*
if *nowTime – startTime > fpsLimitStream*
Run *inference;*
Set *startTime;*
ret, frame(next);

Evaluation metrics

The model's performance is evaluated using mAP (mean average precision), which is a standard metric for measuring the performance of an object detection model. mAP is defined as:

$$mAP = \frac{\sum_{q=1}^{Q} AveP(q)}{Q} \tag{5}$$

Where *Q* is the number of queries in the set and *AveP(q)* is the average precision (*AP*) for a given query *q*.

The mAP is calculated on the bounding box locations for the final two checkpoints. Two Intersection over Union (IOU) thresholds, 0.50 and 0.75, are used to assess the overall performance of the model. The IOU is

[4] www.conservationai.co.uk

a useful metric for determining the accuracy of the bounding box location. This is achieved by measuring the percentage ratio of the overlap between the predicted bounding box and the ground truth bounding box (Lee et al. 2017). A threshold of 0.50 measures the overall detection accuracy while the upper threshold of 0.75 measures localisation accuracy. Figure 4 shows an example prediction during the model training where the ground truth is shown on the right-hand side and the predicted bounding box and label are shown on the left-hand side.

Sensitivity (Sens) and Specificity (Spec) are implemented to describe the correctly classified objects using a one verse all approach. Accuracy (ACC) is used to determine the overall percentage of correct classifications. Confidence intervals (CI) are used to quantify the uncertainty of an estimate based on asymptotic normal approximation. A 95% confidence level shows the likelihood that the range x to y covers the true Sensitivity, Specificity and ACC values of a particular model.

To assess the performance of the frame sampling technique and determine the optimal configuration for the underlying GPU, the following processing metrics are used:

- **Decode Setting (DC)** describes the total number of frames to be analysed within a specified time period. The coefficient value can be set between 1 and 0.0001 therefore controlling the number of frames to be inferenced. The higher end range reduces the number of frames serialised for inferencing, which increases playback speed. The lower end increases the number of frames to inference therefore decreasing the playback speed. The trade-off between the two metrics is application specific.
- **Video Frame Rate** specifies the frequency rate of the consecutively captured frames from the video source.
- **Total Video Frames** provides the total amount of frames transmitted from the feed based on current playback time.
- **Total Video Frames Analysed (TVFA)** is the number of frames processed for inferencing within the total duration of the video. This metric is calculated in algorithm 1 by counting each of the frames submitted to the object detection model.
- **Percentage of Frames Analysed (PFA)** is the TVFA x 100 / Total Video Frames.
- **Runtime(s)** indicates how long the framework took to process all of the analysed frames (TVFA).
- **Total Video Time(s)** is the total length of the video.
- **Processed Frames per Second (PFPS)** is the Total Video Frames divided by the run time.

RESULTS

In this section, the performance of the model and the associated frame sampling technique is evaluated using the metrics outlined in the methodology section. The first experiment reports the results from training the model using the data previously described in the methodology section. This provides standard object detection performance metrics (mAP, IOU) for the trained model used throughout the practical experimentation in this section. The second experiment evaluates the models inferencing capabilities using the trained model and unseen test data from YouTube. Empirically the model training and inference was tuned to provide a viable model and platform for field trials conducted at Knowsley Safari. In the Knowsley Safari experiment we evaluate the model's ability to perform image classification on single objects and object detection for the purpose of object counting.

Training Results

The first experiment empirically evaluates the different network configuration and hyperparameter settings. The results in this section describe the best configuration from the experimentation. Model training was performed over 3457 epochs to minimise the cost function through backpropagation. The total loss at epoch 3457 was 0.6539. This increased slightly from 0.6395 at epoch 3071 suggesting an overshot of the global minimum. Figure 5a shows the total loss for the training session on the y-axis and the epochs on the x-axis. The mAP is evaluated for each bounding box location for the final two checkpoints (epochs 3071 and 3457) respectively. At epoch 3457, the model attained a mAP @.50IOU of 0.83 and 0.69 @.75IOU. At epoch 3017 the mAP was 0.80 @.50IOU and 0.68 @.75IOU. The performance of the model is highlighted in Figures 5b and 5c where the mAP is shown on the y-axis and the epochs on the x-axis.

Inferencing Results

The second experiment evaluates the trained model's ability to generalise and classify on unseen data and evaluate the speed of inference using different configurations. To assess the frame sampling technique and to determine the underlying performance of the GPU architecture a test video was uploaded from YouTube to the object detection pipeline[5]. The video used to test the framework had a total video time of 37 seconds consisting of 925 frames at a resolution of 1280 x 720. The frame rate was set at 25fps. The same video from YouTube was used to baseline the performance of the underlying hardware before real-time inferencing was conducted.

Using the video configuration previously discussed, several tests with videos found on YouTube[6] were performed to evaluate the precision/speed trade-off and overall video classification performance. These metrics allow us to specify the necessary hardware requirements and overall compute time for live inferencing in production-ready systems, as well as those for offline video processing tasks. Table 2 presents the results for the frame sampling technique.

Referencing the results in Table 2, empirically the optimal coefficients value is 0.05. This allows us to process 103 frames out of the original 925 frames (approximately 11% of the total frames). This provides sufficient frame processing and synchronisation with the live video (no notable lag is detected). To demonstrate the configuration, a field trial was conducted at Knowsley Safari.

Knowsley Safari Field Trial

In the third and final experiment the trained model and inference configuration obtained from the previous experiments were used in the Knowsley Safari field trial to detect rhinos and cars from a drone. A DJI Mavic Pro 2 was used to stream 4k video at 30fps of the rhino enclosure at Knowsley Safari from an altitude of 60m. Using the decode setting (DC) 0.05 we were able to conduct object detection at 2fps while maintaining synchronisation with the live video stream and ensuring detection boxes remain fixed around objects. The inferencing was undertaken on the online conservation AI server previously described. The frame sampling technique is the mechanism allowing us to maintain real-time synchronisation with the live video stream whilst conducting live inference. Figure 6 shows the detection of rhinos and cars within the enclosure.

In total 11% (843) of the frames collected were analysed over a 10-minute flight period. This equates to approximately 2fps. If we were to run this for the duration of the battery life of the DJI Mavic Pro 2 the

[5] https://www.youtube.com/watch?v=AJZmL12KNzg

[6] https://www.youtube.com/watch?v=jigx0x4B938

https://www.youtube.com/watch?v=4WuhqZtfyZo

https://www.youtube.com/watch?v=ah02hlhtFdU

system on the specified hardware is capable of processing approximately 2000 images. Note that the two objects detected within figure 6 include rhinos and cars.

Knowsley Safari Field Trial Model Performance for Single Class Detections

In the Knowsley Safari field trial the model was evaluated using 500 images of rhinos and 500 images containing no rhinos. The results of this experiment are shown on row 1 of Table 3. This demonstrates that the model can detect rhinos with 91% sensitivity. The results are slightly lower for images that contain no rhinos with 78% for specificity. In addition, we evaluated the model using 500 images of cars and the same 500 images which did not contain a car. The results are shown in Table 3 row 2, where the model successfully classified all images containing a car with a sensitivity value of 100%. No false detections were made in the 500 images containing no cars, with a specificity value of 100%.

Knowsley Safari Field Trial Model Performance for Class Counting

In the previous experiment, the model was only evaluated on the detection of a single rhino or car within the image (for example does the frame contain a class of interest). In this experiment, we were also interested to see how many rhinos and cars could be detected in each image in a real-world setting (i.e., images containing multiple instances of the same class for the purpose of counting the number of distinct classes in a frame). In order to assess the performance of the trained model, the research team randomly selected 200 frames from the 843 frames acquired from the live feed. The total number of rhino objects in the subsample equates to 591, while the total number of car objects equates to 874. This was done by counting each individual object in each frame. The model successfully detected a total of 425 rhino objects and 638 car objects. For the rhino class, the overall accuracy was 72% (95% CI:68%,76%) while the car class achieved an overall accuracy of 73% (95% CI:70%,76%).

DISCUSSION

The aim of this study was to explore the workflow of a near-real time object detection system for animals and cars using a consumer-grade drone connected to a data centre through the GSM network using a field laptop, to analyse the returned results. The alternative approach that we are aware of that uses the GSM network for linking the live stream from a drone to a data centre in a conservation setting, using a bespoke drone system with a radio link, to transmit the data from the camera to the ground control station is Bondi et. al. (Bondi et al 2018). We are not aware of studies using a field laptop connected to a consumer-grade drone in a conservation setting to conduct object detection using Faster-RCNN in near real time.

The results of this approach are promising. Firstly, a limited set of 350 images per class were used to train our model. The images in the dataset contained large amounts of variance, representing both distant and close-up shots, as well as thermal and grey scale representations. This is not widely reported in many studies. The mAP and IOU are therefore competitive. Figures 5 shows a mAP @.50IOU of 0.83 and 0.69 @.75IOU for the trained model. Secondly, the strategic use of transfer learning shows that the detailed features provided in the Faster-RCNN model can be combined with those extracted from our images to detect new kinds of objects (in this case rhinos). Training on the 350 images without transfer learning would not support the mAP and IOU results produced in this paper. Lastly, using Faster-RCNN and high-quality images without the need to downsample improved detection at higher altitudes.

One of the main novel contributions outlined in this paper is the framework's ability to maximise precision and throughput while maintaining video detection box synchronisation. This provides a unique platform for integrating systems with different hardware and computational capabilities. These range from high-end compute infrastructures to laptops deployed in the field. This allows the regulation of video frames and provides processing support to a wide variety of camera systems such as a drone but also camera traps. The results presented in Table 2 provide a clear metric between the desired number of frames to analyse verses the compute time needed. For example, a DC of 0.05 facilitates the processing of approximately 11% of

captured frames using the trained model. In the example shown in Table 2, a 37-second video containing 925 frames would make it possible to process 103 frames over 45 seconds. In contrast, processing 482 (roughly half) would take 187 seconds. The results presented in Table 2 were achieved using a NVidia Quadro RTX 8000 GPU costing roughly $10,000. Note that we use the NVidia Quadro RTX 8000 GPU for hosting multiple parallel services which is not possible using non-enterprise GPU's.

The feasibility of the system was evaluated in a field trial at Knowsley Safari. Our initial test demonstrated the real-time inferencing capabilities of our approach. Due to poor GSM coverage, the DC had to be set at 0.1 to allow us to process 6% of the total frames acquired. Nonetheless, due to the flexibility of the framework, we were also able to switch to local inferencing using an on-board RTMP server and a laptop containing a NVidia 1050Ti GPU located in the field; we deliberately selected a low-end GPU to demonstrate the accessibility of the system. This allowed us to set the DC back to 0.05 while maintaining video synchronisation.

CONCLUSIONS AND FUTURE WORK

Many conservation applications require manual and time-consuming tasks such as reviewing video footage for animal detection, poacher detection, and animal density estimation. The approach described in this paper provides a possible tool to alleviate the burden of manual routine tasks in conservation by using consumer-grade drones to collect and transmit data to ML models for near real-time image analysis. This can be achieved using cellular networks or ad-hoc hotspots to devices located in the field. Although for some applications the 7km video transmission range might be a limiting factor when needing to monitor large areas, there should be a range of applications for which 7km is sufficient. Due to the relatively low cost of this system (a consumer-grade drone and a field laptop (costing roughly $3000)) the system is accessible to conservation teams. Our experiment used rhinos and cars as an example, but the network could be trained on other animals and objects. Therefore, this technology has significant potential to help in a number of animal survey applications. The current system provides a first step prototype, however, there are several challenges which need to be addressed. Firstly, whilst we trained our study of 350 images per class, deep learning models would need much larger dataset to generalise to new environments. Obtaining a large corpus of images for animals is a significant challenge given that it is often difficult to gain direct access to them in their natural environment. For these reasons we felt it necessary to try and offset this limitation using realistic video footage found on YouTube. It is important to note that this option will not be available for all species. Secondly, the system was trialled in a safari park to test it in a near realistic environment. However, we recognise that this is no match for observing animals in their natural habitat and may degrade the performance of the system.

The proposed baseline solution will be extended in future work. Concentrated efforts will focus on developing the system further to support complex conservation tasks, of which one of the key factors will be scalability and cost. Furthermore, the processing of more complex images is seen as a vital component for better understanding animals and the unique behaviours they exhibit. Along with advancements in camera and drone technology, this will allow for the monitoring of animals at much higher flying heights, thus helping to minimise disruption to animals in their natural habitats. We envision that the data generated will provide a live feed to advanced data analytic dashboards such as PowerBi to provide detailed insights into changes in animal and environmental behaviours. In addition, work will be undertaken to include a broader representation of endangered species in object detection tasks.

The overarching aim of object detection is to detect and derive a more in-depth understanding of animals and their natural habitats. This includes the many aspects which impact on their survival, such as poaching and deforestation. By understanding what happens between successive time points we can build an understanding of the objective differences, and design interventions to mitigate negative effects. In order to

achieve this, a great emphasis on object detection (and in future, advanced analytics on density for instance) is required. This paper provides the first step in this direction.

ACKNOWLEDGEMENTS

The authors would like to thank Naomi Davies at Knowsley Safari for co-ordinating the field trial and providing access to the rhino enclosure. This research was supported by the Research Council UK (RCUK) Science and Technology Facilities Council (STFC) through grant ST/R002673/1.

REFERENCES

Ba, J., and Frey, B. 2013. 'Adaptive dropout for training deep neural networks': 3084--92.

Banerjee, D.S., Hamidouche, K., and Panda, D.K. 2016. 'Re-designing CNTK deep learning framework on modern GPU enabled clusters': 144--51.

Bondi, E., Fang, F., Hamilton, M., Kar, D., Dmello, D., Choi, J., Hannaford, R., Iyer, A., Joppa, L., and Tambe, M. 2018. "Spot poachers in action: Augmenting conservation drones with automatic detection in near real time." In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Bondi, E., Fang, F., Hamilton, M., Kar, D., Dmello, D., Noronha, V., Choi, J., Hannaford, R., Iyer, A., and Joppa, L. 2019. 'Automatic Detection of Poachers and Wildlife with UAVs', *Artificial Intelligence and Conservation*: 77.

Buckland, S. T., Anderson, D.R., Burnham, K.P., Laake, J.L., Borchers, D.L., and Thomas, L. 2001. *Introduction to Distance Sampling* (Oxford University Press: Oxford).

———. 2004. *Advanced Distance Sampling: Estimating Abundance of Biological Populations* (Oxford University Press: Oxford).

Chabot, D., and Bird, D.M. 2015. 'Wildlife research and management methods in the 21st century: Where do unmanned aircraft fit in?', *Journal of Unmanned Vehicle Systems*, 3: 137-55.

Christie, K.S., Gilbert, S.L., Brown, C. L., Hatfield, M., and Hanson, L. 2016. 'Unmanned aircraft systems in wildlife research: current and future applications of a transformative technology', *Frontiers in Ecology and the Environment*, 14: 241-51.

Crunchant, A-S., Borchers, D., Kühl, H., and Piel, A. 2020. 'Listening and watching: Do camera traps or acoustic sensors more efficiently detect wild chimpanzees in an open habitat?', *Methods in Ecology and Evolution*, 11: 542-52.

Fang, Y., Du, S., Abdoola, R., Djouani, K., and Richards, C. 2016. 'Motion Based Animal Detection in Aerial Videos', *Procedia Computer Science*, 92: 13-17.

Hazelwood, K., Bird, S., Brooks, D., Chintala, S., Diril, U., Dzhulgakov, D., Fawzy, M., Jia, B., Jia, Y., Kalro, A., et al. 2018. 'Applied machine learning at facebook: A datacenter infrastructure perspective': 620--29.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. 2017. 'Gans trained by a two time-scale update rule converge to a local nash equilibrium': 6626--37.

Jakobs, S., Weber, A., and Stapp, D. 2019. 'Reliable Object Detection for Autonomous Mobile Machines', *ATZheavy duty worldwide*, 12: 44--49.

Kanai, S., Fujiwara, Y., and Iwamura, S. 2017. 'Preventing gradient explosions in gated recurrent units': 435--44.

Kingma, D.P., and Ba, J. 2014. 'Adam: A method for stochastic optimization', *arXiv preprint arXiv:1412.6980*.

Lamba, A., Cassey, P., Segaran, R.R., and Pin Koh, L. 2019. 'Deep learning for environmental conservation', *Current Biology*, 29: R977-R82.

LeCun, Y., Haffner, P., Bottou, L., and Bengio, Y. 1999. 'Object recognition with gradient-based learning.' in, *Shape, contour and grouping in computer vision* (Springer).

Lee, J., Wang, J., Crandall, D., and Šabanović, S., and Fox, G. 2017. 'Real-time, cloud-based object detection for unmanned aerial vehicles': 2017 First IEEE International Conference on Robotic Computing (IRC) 36--43.

Lim, K., Hong, Y., Choi, Y., and Byun, H. 2017. 'Real-time traffic sign recognition based on a general purpose GPU and deep-learning', *PLOS ONE*, 12.

Longmore, S.N., Collins, R.P., Pfeifer, S., Fox, S.E., Mulero-Pazmany, M., Bezombes, F., Goodwin, A., Ovelar, M., Knapen, J.H., and Wich, S.A. 2017. 'Adapting astronomical source detection software to help detect animals in thermal images obtained by unmanned aerial systems', *International Journal of Remote Sensing*, 38: 2623-38.

Maire, F., Mejias Alvarez, L., and Hodgson, A. 2015. "Automating marine mammal detection in aerial images captured during wildlife surveys: A deep learning approach." In *Australasian Joint Conference on Artificial Intelligence*, 379-85. Springer.

Martinez, P., Ahmad, R., and Al-Hussein, M. 2019. 'A vision-based system for pre-inspection of steel frame manufacturing', *Automation in Construction*, 97: 151--63.

Maxwell, S.L., Fuller, R.A., Brooks, T.M., and Watson, J.E.M. 2016. 'Biodiversity: The ravages of guns, nets and bulldozers', *Nature*, 536: 143-45.

Nichols, J.D, and Williams, B.K. 2006. 'Monitoring for conservation', *Trends in Ecology & Evolution*, 21: 668-73.

Peng, J., Wang, D., Liao, X., Shao, Q., Sun, Z., Yue, H., and Ye, H. 2020. 'Wild animal survey using UAS imagery and deep learning: modified Faster R-CNN for kiang detection in Tibetan Plateau', *ISPRS Journal of Photogrammetry and Remote Sensing*, 169: 364-76.

Rampasek, L., and Goldenberg, A. 2016. 'Tensorflow: Biology's gateway to deep learning?', *Cell systems*, 2: 12--14.

Saria, S., Butte, A., and Sheikh, A. 2018. "Better medicine through machine learning: What's real, and what's artificial?" Open Access. Available at: https://doi.org/10.1371/journal.pmed.1002721

Shin, H-C., Roth, H.R., Gao, M., Lu, Le, Xu, Z., Nogues, I., Yao, J., Mollura, D., and Summers, R.M. 2016. 'Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning', *IEEE transactions on medical imaging*, 35: 1285--98.

Talukdar, J., Gupta, S., Rajpura, P.S., and Hegde, R.S. 2018. 'Transfer learning for object detection using state-of-the-art deep neural networks': 78--83.

van Gemert, J.C., Verschoor, C.R., Mettes, P., Epema, K., Pin Koh, L., and Wich, S.A. 2014. 'Nature Conservation Drones for Automatic Localization and Counting of Animals'.

Wich, S.A., and Pin Koh, L. 2018. *Conservation Drones* (Oxford University Press).

Table 1. Data Collection Sources Per Class

Image Source	Number of RGB	Number of Thermal	Total Number of Sampled Images
Drone based camera	140	35	175
Ground based camera	21	64	85
Google web search	90	90	90

Table 2. Frame Processing Results

DC	**% PFA**	**Runtime (s)**	**PFPS**	**TVFA**
1	0.1	3	308	5
0.1	5.9	21	44	55
0.08	7.1	29	31	66
0.07	8.6	42	22	80
0.06	10.07	32	28.9	99
0.05	11.13	45	20.55	103
0.01	52	154	6	482
0.001	53	170	5.4	480
0.0001	52	187	4.9	482

Table 3. Model testing results

Class	Sens (95% CI)	Spec (95% CI)	ACC (95% CI)
Rhino	0.91(0.869,094)	0.78(0.74,0.82)	0.84(0.81,0.87)
Car	1.00(1.00,1.00)	1.00(1.00,1.00)	1.00(1.00,1.00)

Figure captions:

Fig. 1: Example training data with variance

Fig. 2: Faster-RCNN Architecture

Fig. 3: Object Detection Pipeline

Fig. 4: Total loss, mAP @.50IOU and mAP @.75IOU

Fig. 5: Knowsley Safari Field Trial

Fig. 6: Detection of rhinos and cars within the Safari Park enclosure

Fig.1

Fig. 2

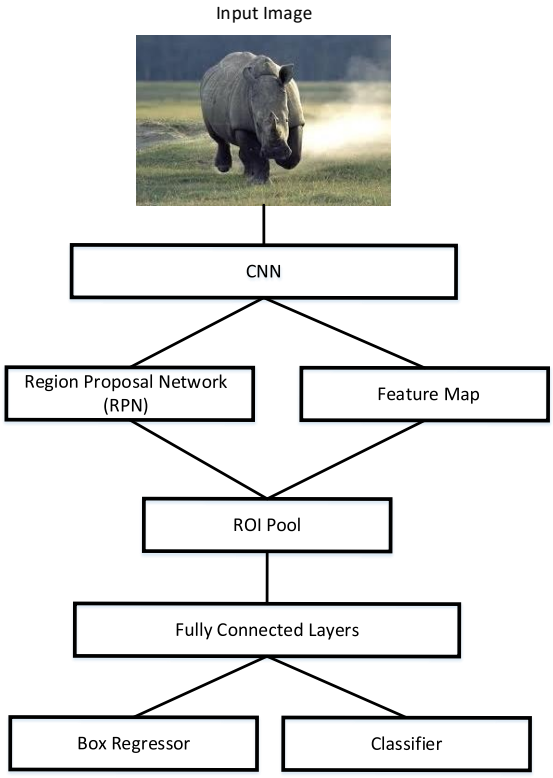

Input Image
CNN
Region Proposal Network (RPN)
Feature Map
ROI Pool
Fully Connected Layers
Box Regressor
Classifier

Fig. 3

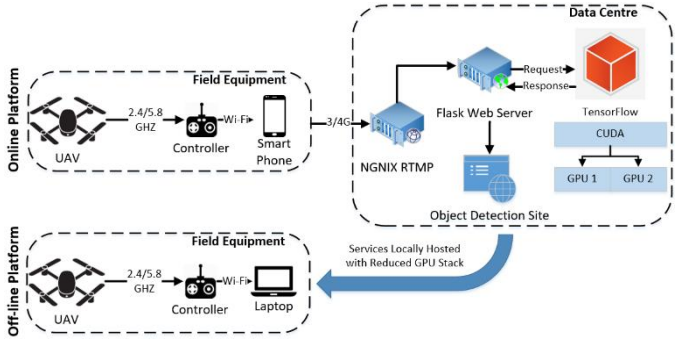

Data Centre
Online Platform
Field Equipment
UAV
2.4/5.8 GHZ
Controller
Wi-Fi
Smart Phone
3/4G
NGNIX RTMP
Flask Web Server
Request
Response
TensorFlow
CUDA
GPU 1
GPU 2
Object Detection Site
Off-line Platform
Field Equipment
UAV
2.4/5.8 GHZ
Controller
Wi-Fi
Laptop
Services Locally Hosted with Reduced GPU Stack

Fig.4

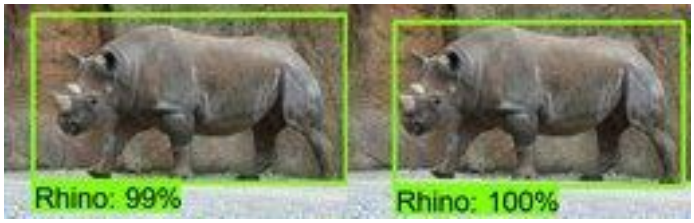
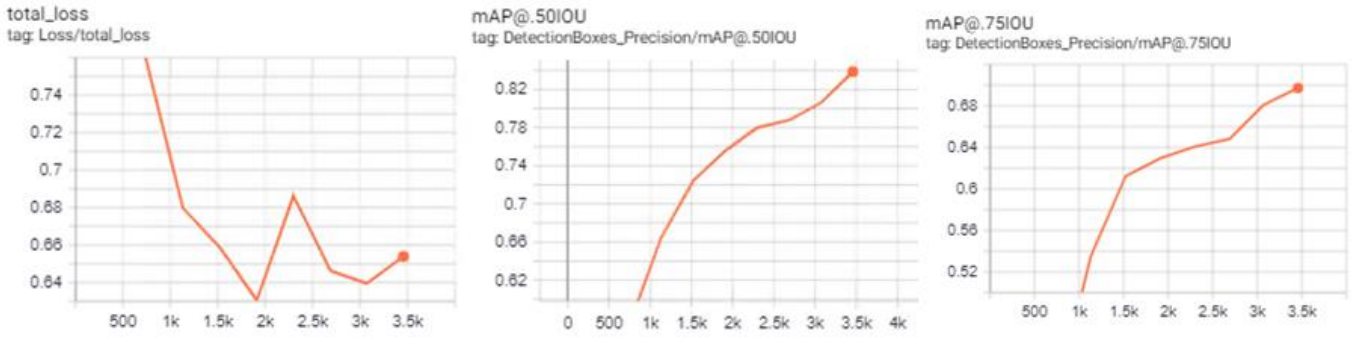
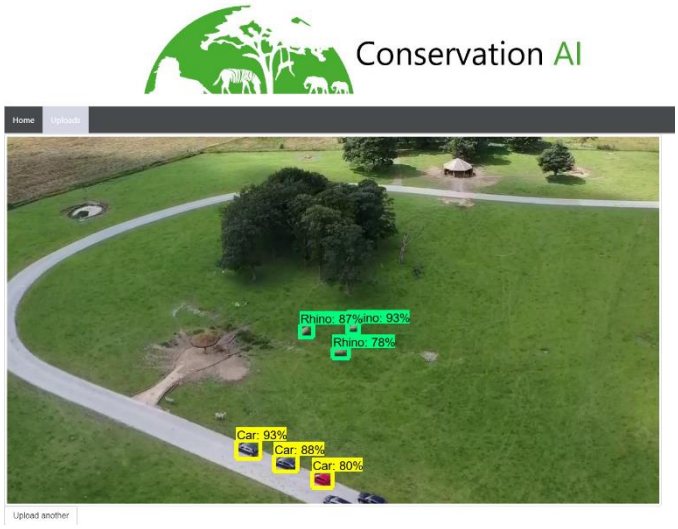

Rhino: 99%
Rhino: 100%

Fig. 5



(a) Total Loss (b) mAP @.50IOU (c) mAP @.75IOU

Fig. 6


Conservation AI
Home
Uploads
Rhino: 87%ino: 93%
Rhino: 78%
Car: 93%
Car: 88%
Car: 80%
Upload another