Animal Detection in Man-made Environments

Abhineet Singh [1], Marcin Pietrasik [2], Gabriell Natha [2], Nehla Ghouaiel [2], Ken Brizel [2], Nilanjan Ray [1]

[1]Department of Computing Science, University of Alberta
[2]Alberta Centre for Advanced MNT Products (ACAMP)

Abstract

Automatic detection of animals that have strayed into human inhabited areas has important security and road safety applications. This paper attempts to solve this problem using deep learning techniques from a variety of computer vision fields including object detection, segmentation, tracking and edge detection. Several interesting insights into transfer learning are elicited while adapting models trained on benchmark datasets for real world deployment. Empirical evidence is presented to demonstrate the inability of detectors to generalize from training images of animals in their natural habitats to deployment scenarios of man-made environments. A solution is also proposed using semi-automated synthetic data generation for domain specific training. Code and data used in the experiments are made available to facilitate further work in this domain.

1. Introduction

Object detection is an important field in computer vision that has seen very rapid improvements in recent years using deep learning [96, 48, 67]. Most detectors are trained and tested on benchmark datasets like COCO [66], Open Images [61], KITTI [33] and VOC [30]. In order to apply these in a particular domain like animal detection, a model pre-trained on one of these datasets is fine-tuned on domain-specific data, usually by training only the last few layers. This is known as transfer learning [73, 115] and is often enough to obtain good performance in the new domain as long as it does not differ drastically from the original. The goal of this work is to use transfer learning to adapt state of the art object detection methods for detecting several types of large Alberta animals in real-time video sequences captured from one or more monocular cameras in moving ground vehicles. The animals that most commonly stray into human habitations include: deer, moose, coyotes, bears, elks, bison, cows and horses. There are two deployment scenarios:

- Detecting threats in an autonomous all-terrain vehicle (ATV) patrolling the Edmonton International Airport perimeter for security and surveillance purposes.
- Finding approaching animals in side–mounted cameras on buses plying the Alberta highways to issue a timely warning to the driver for collision avoidance.

The main challenge here is the scarcity of existing labeled data with sufficient specificity to the target domain to yield good models by fine-tuning pre-trained detection networks. Although several of the large public datasets like COCO [66] do include some of the more common animals like bears and horses, these rarely include the Canadian varieties that are the focus of this work and often feature incorrect backgrounds. Even larger classification datasets like Imagenet [27] do include images of many of the target animals but only provide image level labels so the bounding boxes would have to be added manually. There are also several animal specific datasets [4, 38] but these likewise do not match the target requirements well, having, for example, aerial viewpoints [57, 58], incorrect species [60, 75, 62, 108, 44, 93, 74] or habitats [13, 12] and no bounding box annotations [45, 93, 98, 111].

The lack of training data was addressed by collecting and labelling a sufficiently large number of images of the target animals. This was initially confined to videos since labeling these was easier to semi-automate (Sec. 3.1) and training detectors on videos showing the animals in a variety of poses seemed to concur better with deployment on camera videos captured from moving vehicles. However, tests showed that detection performance is far more sensitive to the range of *backgrounds* present in the training set rather than variations in the appearance of the animal itself (Sec. 4). Though static images helped to resolve this to a certain extent, they are much harder to obtain in large numbers and a lot more time-consuming to label. More importantly, neither static nor video images of animals are easy to acquire with the kinds of structured man-made surroundings that the airport perimeter and highways present. This paper thus proposes a solution based on synthetic data generation using a combination of interactive mask labelling, instance segmentation and automatic mask generation (Sec. 3.4).

Another significant challenge is the need for the detector to be fast enough to process streams from up to 4 cameras in real time while running on relatively low-power machines

since both deployment scenarios involve mobile computation where limited power availability makes it impractical to run a multi-GPU system. This is addressed using RetinaNet [65] and YOLOv3 [82] which turned out to be surprisingly competitive with respect to much slower models. To the best of our knowledge, this is also the first large-scale study of applying deep learning for animal detection in general and their Canadian varieties in particular. It presents interesting insights about transfer learning gained by training and testing the models on static, video and synthetic images in a large variety of configurations. Finally, it provides practical tips that might be useful for real world deployment of deep learning models. Code and data are made publicly available to facilitate further work in this field [10].

2. Related Work

Animal recognition in natural images is a well researched area with applications mostly in ecological conservation. As in the case of available data, most of the existing work is not closely allied to the domain investigated in this paper. Three main categories of methods can be distinguished from the literature corresponding to the type of input images used. The first category corresponds to aerial images captured from unmanned aerial vehicles (UAVs). A recent work [56] introduced an active learning [101] method called transfer sampling that uses optimal transport [25] to handle domain shift between training and testing images that occurs when using training data from previous years for target re-acquisition in follow-up years. This scenario is somewhat similar to the current work so transfer sampling might have been useful here but most of this work had already been done by the time [56] became available. Further, it would need to be reimplemented since its code is not released and the considerable domain difference between aerial and ground imagery is likely to make adaptation difficult. Finally, most domain adaptation methods, including [56], require unlabeled samples from the target domain which are not available in the current case. Other examples of animal detection in UAV images include [59, 85, 58, 57] but, like [56], all of these are focused on African animals.

The second category corresponds to motion triggered camera trap images. These have been reviewed in [91] and [14] where the latter reported similar difficulties in generalizing to new environments as were found here. The earliest work using deep learning was [20] where graph cut based video segmentation is first used to extract the animal as a moving foreground object and then a classification network is run on the extracted patch. A more recent work [90] that most closely resembles ours tested two detectors - Faster RCNN [83] and YOLOv2 [81] - and reported respective accuracies of 93% and 76%. However, the evaluation criterion used there is more like classification than detection since it involves computing the overlaps of all detected boxes with the ground truth and then comparing the class of only the maximum overlap detection to decide if it is correct. Other recent works in this category, most of them likewise dealing mainly with classification, include [72, 111, 98, 117, 116, 119].

Table 1: Annotation counts (**seq**, **syn**: sequences, synthetic)

Class	Videos (Real)		Static Images			Total
	Seq	Images	Real	Syn	Total	
Bear	92	25715	1115	286	1401	27116
Bison	88	25133	0	0	0	25133
Cow	14	5221	0	0	0	5221
Coyote	113	23334	1736	260	1996	25330
Deer	67	23985	1549	286	1835	25820
Elk	78	25059	0	0	0	25059
Horse	23	4871	0	0	0	4871
Moose	97	24800	0	260	260	25060
Total	**572**	**158118**	**4400**	**1092**	**5492**	**163610**

The third category, which includes this work, involves real-time videos captured using ground-level cameras. An important application of such methods is in ethology for which many general purpose end-to-end graphical interface systems have been developed [71, 92, 109, 86, 76]. Methods specialized for particular species like cows [124], beef cattle [99] and tigers [62] have also been proposed where the latter includes re-identification that is typically done using camera trap images. Surveillance and road safety applications like ours are much rarer in the literature and it seems more common to employ non-vision sensors and fencing/barrier based solutions, probably because many animal vehicle collisions happen in the dark [110]. Examples include infrared images [32], thermal and motion sensors [37], ultra wide band wireless sensor network [113] and kinect [120].

A general review of the vision techniques used in this work including object detection, segmentation and tracking are excluded here due to space constraints. The actual methods used in the experiments are detailed in Sec. 3.

3. Methodology

3.1. Data Collection

To facilitate the large number of training images needed, a combination of video and static images was used. Video was collected both directly with handheld video cameras around Calgary area, such as the Calgary Zoo, as well as online via YouTube and Nature Footage [5]. Due to the large quantity of static images that was required, downloading them one by one was not feasible. Instead, ImageNet [27] was used as it provides a searchable database of images with links whereby they can be downloaded in bulk using scripts. However, not all animal species are available there and not all available ones have enough images. Google Images was thus also used by searching for specific taxonomic classification and downloading the results in bulk using browser extensions. After downloading static images, it was neces-

Figure 1: Sample collected images: (clockwise from top left) bear (Calgary Zoo), deer (Google Images), coyote (Google Images), elk (Nature Footage), horse (YouTube), cow (YouTube), moose (YouTube) and bison (Calgary Zoo)

sary to verify that all were of the intended animal and remove any mislabeled or unhelpful images. Figure 1 shows sample images of all animals while Table 1 provides quantitative details.

3.2. Labeling

3.2.1 Bounding Boxes

Annotation was done using a heavily modified version of an open source image annotation tool called LabelImg [102]. This tool takes a video file or sequence of images as input and allows the users to annotate bounding boxes with class labels over them. The SiamFC tracker [17] was integrated with the tool to make video annotation semi–automated so that the user only needs to manually annotate the animal in the first frame, track it till the tracker starts drifting, fix the box in the last tracked frame, start tracking again and repeat this process till all frames are labeled.

3.2.2 Segmentation Masks

Pixel wise masks were needed to generate high-quality synthetic data (Sec. 3.4). Annotation tools that support masks do exist [29, 28, 103, 19, 88, 55, 3], including AI assisted services [6], but all have issues such as too course masks [29, 28, 103, 19], Linux incompatibility [55], paid or propriety license [6, 3] or cloud-data restriction [88]. Also, it was desirable to semi-automate mask generation using the already existing bounding boxes which is not allowed by any of the tools. Mask annotation functionality was thus added to the labelling tool with support for 3 different modalities to add or refine masks - drawing, clicking to add boundary points and painting with variable sized brushes.

Semi-automated mask generation was done using a combination of motion based interpolation, edge detection and tracking. An approximate mask is generated for a given frame by estimating the motion between its bounding box and that in a previous frame whose mask has already been annotated. In addition, holistically nested edge detection (HED) [112] followed by adaptive thresholding is used to obtain a rough boundary of the animal that can be refined by painting. Finally, the SiamMask tracker [106], that outputs both bounding boxes and segmentation masks, was integrated with the labelling tool to generate low-quality masks in a fully automated manner. Mask labelling was a slow and laborious task and took anywhere from 1 - 8 minutes per frame depending on animal shape and background clutter. An arguably more sophisticated pipeline for rapid generation of segmentation masks has recently been proposed [15]. However, it became available too late to be utilized in this project, does not provide a publicly available implementation and its proposed pipeline includes human involvement on too large a scale to be practicable here. A recent video mask prediction method [114] likewise came out too late and also rendered unnecessary by SiamMask.

3.3. Object Detection

Object detection has improved considerably since the advent of deep learning [96] within which two main categories of detectors have been developed. The first category includes methods based on the RCNN architecture [35, 36] that utilize a two-step approach. A region proposal method is first used to generate a large number of class agnostic bounding boxes that show high probability of containing a foreground object. Some of these are then processed by a classification and regression network to give the final detections. Examples include Fast [34] and Faster [83, 84] RCNN and RFCN [26]. The second category includes methods that combine the two steps into a single end to end trainable network. Examples include YOLO [80, 81, 82], SSD [70] and RetinaNet [64, 65]. Apart from its high-

Figure 2: Synthetic data samples with corresponding source and target images for (top) coyote on airport and (bottom) moose on highway. Each row shows (left to right) animal source image, target background image and crops of synthetic images generated using (clockwise from top left) manual labeling, Mask RCNN, SiamMask and Gaussian blending (no mask).

level architecture, the performance of a detector also depends on the backbone network used for feature extraction. Three of the most widely used families of performance-oriented backbones include ResNet [42, 43, 51], Inception [95, 97, 24, 94] and Neural Architecture Search (NAS) [125, 126]. Several architectures have also been developed with focus on high speed and low computational requirements. The most widely used among these are the several variants of MobileNet [47, 89, 46].

Five high level detector architectures have been used here – Faster RCNN, RFCN, SSD, RetinaNet and YOLO. Three different backbone networks are used for Faster RCNN - ResNet101, InceptionResnetv2, NAS - and two for SSD - Inceptionv2 [97], Mobilenetv2 [89] - for a total of 8 detectors. ResNet101 and ResNet50 are used as backbones for RFCN and RetinaNet respectively. All 3 variants of YOLO [80, 81, 82] were experimented with, though only YOLOv3 [82] results are included here as being the best performer. These methods were chosen to cover a good range of accuracies and speeds among modern detectors.

All of the above are *static* detectors that process each frame individually without utilizing the temporal correlation inherent in video frames. Detectors have also been developed to incorporate this information for reducing missed detections due to issues like partial occlusions and motion blur. Examples include Seq-NMS [40], TCNN [54, 53], TPN [52], D&T [31] and FGFA [121, 122]. However, none of these have compatible implementations and most need either optical flow, patch tracking or both to run in parallel with a static detector which makes them too slow to be used here. LSTM-SSD [68, 69] is the only recent video detector that is both fast and open source but attempts to incorporate this here showed its implementation [8] to be too buggy and poorly documented to be usable without significant reimplementation effort not warranted by the modest improvement it seemed likely to provide. Instead, a simple algorithm was devised to combine the DASiamRPN tracker [123] with YOLO (Sec. 4.2.6) to gauge the potential benefit of temporal information in videos.

3.4. Synthetic Data Generation

Experiments showed that detectors have limited ability to generalize to new backgrounds (Sec. 4.2.1). A solution considered first was to collect static images with as much background variation as possible to cover all target scenarios. This proved to be impracticable due the difficulty of finding and labeling sufficient quantities of static images, exacerbated by our target scenarios consisting of man-made environments where it is extremely rare to find animals at all. As a result, synthetic data was generated by extracting animals from existing labeled images and adding them to images of the target backgrounds. Attempts were initially made to do this without masks by selecting only the best matching source images for each target background through techniques like histogram matching and then using Gaussian blending to smoothen the transition from source to target background. However, this failed to generate images that could either be perceived as realistic by humans or improve detection performance (Sec. 4.3). Pixel wise masks were therefore generated by manually labelling a sparse collection of frames with as much background variation as possible and then training instance segmentation models (Sec. 3.5) to automatically generate masks for remaining frames with similar backgrounds. SiamMask tracker [106] was also

Table 2: Implementations used for the various methods (TF: Tensorflow, PT: PyTorch)

Methods	Implementations
All static detectors except YOLO	TF (Object Detection API) [9]
YOLOv3, YOLOv2, YOLOv1	PT [50], TF [100], Darknet [79]
Mask RCNN, Sharpmask, FCIS	TF [9], TF [7], MXNet [2]
SiamFC, SiamMask, DASiamRPN	TF [16], PT [105], PT [104]
Deeplab, UNet/SegNet, HED	TF [22], Keras [39], OpenCV[18]

used towards the end of the project to make this process fully automated. Generating synthetic images was much faster than labelling masks and only took about 1-10 seconds/frame. Most of the effort was focused on generating static images since experiments (Sec. 4.2.2) showed that videos do not help to improve detectors much. It is also significantly harder to generate realistic videos as that requires camera motion in the source and target video clips to be identical. Images were generated from 14 airport and 12 highway backgrounds with 11 source images for bears and deer, and 10 for coyotes and moose. Fig. 2 shows examples.

3.5. Instance Segmentation

Instance segmentation distinguishes between each instance of an object as opposed to semantic segmentation that only identifies categories of objects. The former intuitively seems more suitable for extracting animal boundaries from bounding boxes since it uses object level reasoning whereas the latter is more oriented towards pixel-level classification. This was confirmed by experiments with several state of the art semantic segmentation methods, including DeepLab [21, 23], UNet [87] and SegNet [11]. All of these generated masks that were too fine-grained to cleanly segment out the animal from its background, instead producing many small groups of background pixels inside the animal and, conversely, animal pixels within the background. Three instance segmentation methods were then considered – SharpMask/DeepMask [78, 77], Mask RCNN [41] and FCIS [63]. Mask RCNN was found to produce the highest quality masks so only its results are included.

3.6. Implementations and Training

Table 2 lists all implementations used here. Training was done by fine tuning models pre-trained on large benchmark datasets – COCO [1] for Mask RCNN and all detectors; ImageNet [27] for Sharpmask and FCIS; ADE20K [118] for Deeplab, UNet and SegNet. HED and all trackers were used directly with pretrained weights without any fine tuning.

In order to avoid class bias while training, number of samples from all classes must be similar [107]. Number of labeled images, however, varies significantly between animals (Table 1), especially when the source type – video or static – is taken into account. Therefore, experiments were done with 3, 4 and 6 classes (Table 3) in addition to all 8 to cover a range of scenarios while maintaining class balance.

4. Results

4.1. Evaluation Metrics

Object detectors are usually evaluated using their **mean average precision (mAP)** [49], defined as the mean, over all classes, of the area under the recall–precision curve for each class. Although a good measure of the overall threshold-independent performance, mAP may not accurately represent deployment accuracy where a single threshold must be chosen. Since mAP considers the variation of recall and precision with threshold *separately for each class*, and this can differ greatly between classes (Fig. 3c), it is more indicative of accuracy when a different threshold can be chosen for each class to optimize the recall-precision characteristics for that class. It is also difficult to interpret mAP to gauge the practical usability of a detector in terms of how likely it is to miss objects or give false detections. This paper therefore proposes another metric obtained by first averaging recall and precision for each threshold over all classes and then taking the **recall–precision (RP)** value at the threshold where the two are equal. This metric is named **mean Recall-Precision (mRP)** and provides a more interpretable measure of performance when using a single threshold for all classes.

Further, this work deals mainly with human-in-the-loop type security applications where detections alert humans to take suitable measures after verification. In such cases, simply detecting an object can be far more crucial than classifying it correctly. For example, when used as an early warning system for bus drivers, misclassification would have little impact on the driver's response as long as the animal is detected early enough. A less stringent evaluation criterion named **class-agnostic Recall-Precision (cRP)** is thus also used that treats all animals as belonging to the same class so that misclassifications are not penalized.

4.2. Real Data

4.2.1 How well do detectors generalize ?

Fig. 3 summarizes the results for several training and testing configurations (Table 4) used to study the generalization ability of detectors in a range of scenarios. These are henceforth referred to by their **numeric IDs** (first column of Table 4) and detectors by **acronyms** (Fig. 3) for brevity.

Fig. 3a gives results for all detectors in #1 - #5. The large difference between #1 and #2 clearly demonstrates the inability of detectors to generalize to unseen backgrounds. Both have 1K video images/class but the latter has these sampled from all sequences to allow training over nearly all backgrounds in the test set while the former does not get any frames from the tested sequences. This is sufficient for the detectors to achieve near perfect mRPs in #2 while giving far poorer performance with only 35-60% mRP in #2. A similar trend is seen, though to a lesser extent, in #3

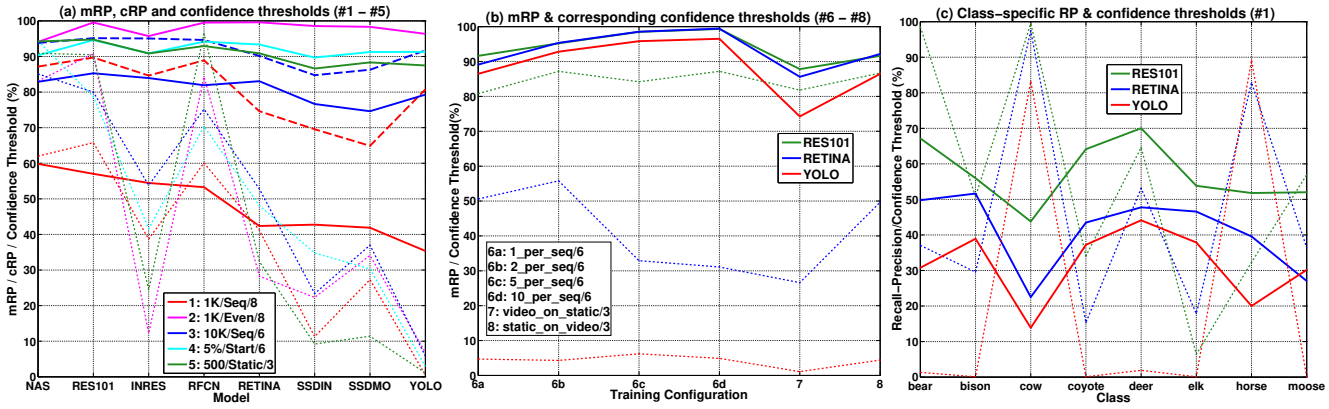

Figure 3: Detection mRP (solid), corresponding confidence thresholds (dotted) and cRP (dashed, only #1, #3): (a) #1 - #5 for all 8 models (b) #6 - #8 for 3 models (c) class-specific #1 results for 3 models. Model Acronyms: **NAS**, **RES101**, **INRES** - Faster RCNN w/ NAS, ResNet101, Inception-ResNetv2 backbones; **RFCN** - RFCN w/ ResNet101; **RETINA** - RetinaNet w/ ResNet50; **SSDIN**, **SSDMO** - SSD w/ Inceptionv2, MobileNetv2; **YOLO** - YOLOv3. Best viewed under high magnification.

and #4. The former, with 10K images/class from complete sequences, is significantly outperformed by the latter with only 5% images from the start of each sequence (or ∼1.2K images/class). The smaller difference here is attributable to the much greater frame count in #3 and the fact that #4 uses consecutive frames from each sequence which contain a smaller range of backgrounds than the evenly sampled frames in #2. Performance in #5 is comparable to #4, even though #5 involves testing over a far greater proportion of unseen backgrounds, probably because most static images depict animals in their natural habitats (Sec. 3.1) which, exhibiting limited variability, allow the detectors to generalize relatively well.

Fig. 3a also shows cRP, though only for #1 and #3 since remaining configurations all had cRP > 90% whose inclusion would have cluttered the plots so these have been deferred to the supplementary. As expected, cRPs are significantly higher than mRPs for all models, though the gain is most notable for YOLO, particularly in #1 where it more than doubles its performance, outperforming both the SSDs as well as RETINA. This suggests, and qualitative examination has confirmed, that the form of overfitting YOLO is susceptible to involves associating backgrounds to specific animals whose training images had similar backgrounds. For example, if a particular scene is present in bear training images but not in those of deer, a test image of a similar scene, but containing deer, would have the animal detected as bear. The other models are susceptible to this too but to a smaller degree and more often miss the animal altogether.

4.2.2 How much are video annotations worth ?

Fig. 3b shows results for #6 - #8; only 3 detectors are included to reduce clutter since the others showed similar performance patterns. #6 involved training with 1, 2, 5 and 10 frames/sequence, with the sequence count limited to 67 by the class with the fewest sequences (deer) to maintain class balance. All 4 models were tested on the same 67 sequences using frames not included in any of their training sets. It can be seen that even 1 frame/sequence is enough for all detectors to give 90% mRP, which improves only marginally with 2 and 5 frames, plateauing thereafter. Further, though RETINA does catch up with RES101 using $\geq$ 2 frames,

Table 3: Class configurations for training (**c**: no. of classes)

c	Animals	Comments
6	all except cow, horse	these have only ∼5K images
4	bear, deer, moose, coyote	synthetic images
3	bear, deer, coyote	real static images

Table 4: Training configurations for both real and synthetic data (**c**, **img**, **seq** - number of classes, images, sequences).

#	c	Details	Train	Test
			img (seq)	img (seq)
1	8	1K video images/class sampled from complete sequences	8001 (33)	150117 (539)
2	8	1K video images/class sampled evenly across all sequences	8156	149962
3	6	10K video images/class sampled from complete sequences	60003 (218)	88023 (317)
4	6	5% images from the start of each video sequence	7169	140857
5	3	500 static images/class	1500	2900
6a-6d	6	1, 2, 5, 10 images sampled evenly from each of 67 sequences	402, 804, 2010,4020	103235
7	3	20K video images/class tested on static images	60000	4400
8a, 8b	3	1K static images/class tested on video, synthetic images	3000	73034, 598
9	4	20K video images/class tested on synthetic images	80008	780
10a, 10b	3, 4	3, 4 class models trained on 28% of synthetic images, tested on rest	234, 312	598, 780

YOLO is unable to even narrow the gap, which might indicate that domain specialization cannot entirely overcome architectural limitations. #7 and #8 show the relative utility of video and static frames by training on one and testing on the other. As expected, static models demonstrate far superior generalizability by outperforming the video models by 4-12% mRP even though the latter are trained and tested on $20\times$ more and $16\times$ fewer frames respectively. Performance gap between #7 and #8 is also larger for worse performing models, especially YOLO that has twice the gap of RETINA, which reaffirms its poor generalizability. Finally, the fact that #8 has lower mRP than #6a even though the former has nearly $15\times$ more images with varied backgrounds shows the importance of domain specialization.

4.2.3 How do the detector accuracies compare ?

RES101 turns out to be the best overall, though NAS, RFCN and INRES remain comparable in all configurations. NAS even has a slight edge in #1, showing its better generalizability under the most difficult scenarios. Conversely, the shortcomings of 1-stage detectors compared to their 2-stage counterparts are also most apparent in #1. This is particularly notable for RETINA that is comparable to RES101 and significantly better than the other 1-stage detectors in all remaining configs. YOLO likewise performs much poorer relative to the two SSDs while being similar and even better in other configs. This might indicate that 1-stage detectors in general, and YOLO in particular, are more prone to overfitting with limited training data. From a practical standpoint, though, YOLO redeems itself well by its relatively high cRPs, outperforming RETINA in both #1 and #3.

4.2.4 How important is the confidence threshold ?

Fig. 3 shows confidence thresholds corresponding to mRP or class-specific RP using dotted lines. Fig. 3c shows that the threshold corresponding to the class-specific RP varies widely between classes - much more than the RP itself. As mentioned in Sec. 4.1, this motivates the use of mRP instead of mAP as a practical evaluation criterion. Further, Fig. 3a,b show that the optimal mRP threshold itself varies greatly between the detectors too. Therefore, choosing a single threshold for all of them might not provide a true picture of their relative performance in practice. It is also evident, especially in Fig. 3b, that a weak correlation exists between the relative performance and threshold, with better performing detectors usually also having higher thresholds. Notable exceptions to this are INRES and SSDIN, both having smaller thresholds than their respective mRP levels. Since both use different variants of Inception, this might be due to an architectural peculiarity thereof. Also notable are the very low thresholds of YOLO - often $< 5\%$ and sometimes even $< 1\%$.

Table 5: Speed, GPU memory consumption and maximum batch size for each detector. Refer Fig. 3 for model names. (Setup: Titan Xp 12GB, Threadripper 1900X, 32GB RAM)

Model	Batch Size 1		Batch Size 4		Max Batch Size	
	memory (MB)	speed (FPS)	memory (MB)	speed (FPS)	batch size	speed (FPS)
NAS	9687	1.36	-	-	3	1.39
INRES	7889	3.95	8145	4.68	8	4.49
RES101	5077	19.61	5589	25.35	36	27.12
RFCN	5041	19.8	5553	32.12	76	26.94
RETINA	4785	31.5	5553	43.51	120	53.28
YOLO	1487	71.41	2039	104.25	48	119.64
SSDIN	3631	68.35	3631	155.63	160	181.66
SSDMO	1999	78.67	2031	167	480	246.56

4.2.5 How resource intensive are the detectors ?

Since both deployment scenarios of ATV and highway buses involve mobile systems with limited power availability, it is important for the detector to be as lightweight as possible. Table 5 shows the speed in frames per second (FPS) along with GPU memory consumption for batch sizes 1 and 4, where the latter is chosen to represent the 4 cameras needed for a simultaneous 360° field-of-view. The maximum batch size that can be run on a 12GB Titan Xp GPU is also shown for scalability comparison. SSDMO turns out to be the fastest, though YOLO is comparable at batch size 1 and also has significantly smaller memory footprint. However, YOLO does not scale as well in either speed or memory and ends up with only a tenth of the maximum batch size of SSDMO and less than half the corresponding speed. NAS and INRES are the slowest and most memory intensive by far and unsuitable for realtime applications. RFCN and RES101 are similar with unit batch size, probably due to their identical backbone, though RFCN scales better, allowing more than twice the maximum batch size and 28% higher speed with batch size 4. Finally, RETINA provides the best compromise between performance and speed - RES101-like mRP in most configs and fast enough to process 4 camera streams simultaneously at 10 FPS each and thus capture an animal visible for a fraction of a second.

4.2.6 Can tracking reduce false negatives ?

As mentioned in Sec. 3.3, tracking was used in an attempt to reduce false negatives by utilizing temporal information in videos. DASiamRPN [123] was used as the tracker as being one of the fastest available Siamese type trackers. YOLO was used as the detector since its PyTorch implementation was easier to integrate with that of DASiamRPN, its speed with batch size 1 (necessary to use tracking) is among the fastest and its poor performance in #1 provides ample scope for improvement. The detailed algorithm is included in the supplementary, though its high level idea is simple - associate detections with existing trackers, create new trackers for unassociated detections and remove trackers that remain unassociated for too long or those with the

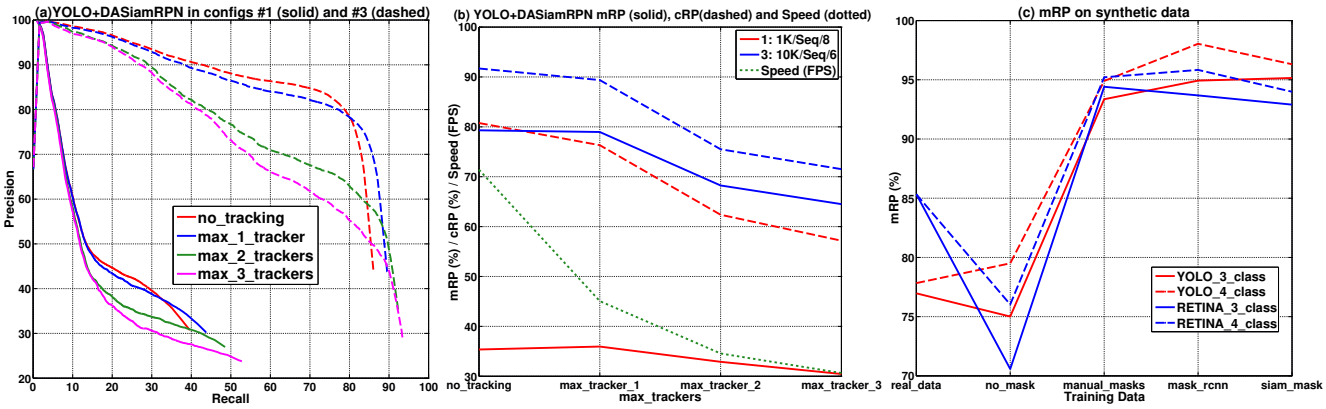

Figure 4: Results for (a - b) DASiamRPN + YOLO and (c) RETINA and YOLO tested on synthetic data

lowest confidence when tracker count exceeds a threshold. Fig. 4a shows the mean Recall vs. Precision plots while Fig. 4b gives mRP / cRP and speeds. Tracking mostly helps only when the detector finds an animal in at least one frame in a sequence and misses it in several subsequent ones. It turns out that this seldom happens in practice so that the resultant increase in recall is very slight and is offset by a significant decrease in precision through tracking false positives. The latter can be mitigated by removing unassociated trackers frequently but this leads to a large drop in recall and is therefore not used in these results. There is thus no net gain in mRP/cRP using tracking, rather significant drops with >1 trackers. When combined with the reduction in FPS, it does not seem like an effective way to reduce false negatives.

4.2.7 Can multi-model pooling reduce false negatives ?

Another way to reduce missing detections is to run multiple detectors simultaneously and pool their detections. A large variety of methods were explored to pool YOLO, SSDIN and SSDMO but none managed to increase recall enough to offset the fall in precision and the net mRPs were even worse than those from tracking. Descriptions of these methods and corresponding results are thus in the supplementary.

4.3. Synthetic data

A training set was constructed from synthetic data by selecting images corresponding to 3 animal poses per background, with a different combination of poses selected randomly for each background, while all remaining images were used for testing. Table 4 denotes the corresponding configs as #10a and #10b for 3 and 4 class models respectively. Corresponding real data configurations are #8b and #9 with 1K static and 20K video images/class respectively. Seperate models were trained for each of the 4 methods of extracting animals from source images (Sec. 3.4) – Gaussian blending, manual masks, Mask RCNN and SiamMask. All were tested on images generated by manual masks.

As shown in Fig. 4c, models trained on synthetic data significantly outperform those trained on real data as long as masks are used. This is remarkable considering that only 78 frames/class were used for the former compared to 1K or 20K for the latter. This reiterates the results in Sec. 4.2.2 where #6a with 67 images outperformed #8a with the same 1K images as #8b. However, unlike there, YOLO does manage to match RETINA here, which suggests that high enough degree of specialization can indeed overcome its architectural shortcomings. More importantly, there is no perceptible difference in mRP between models corresponding to the three segmentation methods. This shows that even the fully unsupervised and visibly coarse masks from SiamMask have comparable detector training information to precise manual masks. At the same time, mask quality does indeed matter since the no mask / Gaussian blending models perform even worse than real data.

5. Conclusions

This paper presented a large scale study of animal detection with deep learning where 8 state of the art detectors were compared in a wide range of configurations. A particular focus of the study was to evaluate their generalization ability when training and test scenarios do not match. It was shown that none of the detectors can generalize well enough to provide usable models for deployment, with missed detections on previously unseen backgrounds being the main issue. Attempts to increase recall using tracking and multi-model pooling proved ineffective. Synthetic data generation using segmentation masks to extract animals from images of natural habitats and inserting them in target scenes was shown to be an effective solution. An almost fully automated way to achieve this was demonstrated by the competitiveness of coarse unsupervised masks with precise manual ones in terms of the performance of detectors trained on the corresponding synthetic images. RETINA and YOLO were shown to be competitive with larger models while being sufficiently lightweight for multi-camera mobile deployment.

References

[1] COCO Dataset. online: http://cocodataset.org/, September 2018. 5

[2] Fully Convolutional Instance-aware Semantic Segmentation. online: https://github.com/msracver/FCIS, September 2019. 5

[3] Labelbox. online: https://labelbox.com/, 2019. 3

[4] Labeled information library of alexandria: Biology and conservation. online: http://lila.science/datasets1, August 2019. 1

[5] Nature footage. https://www.naturefootage.com, 2019. 2

[6] Playment. online: https://playment.io, 2019. 3

[7] TensorFlow implementation of DeepMask and SharpMask. online: https://github.com/aby2s/sharpmask, September 2019. 5

[8] Tensorflow Mobile Video Object Detection. Github, 2019. https://github.com/tensorflow/models/research/lstm_object_detection. 4

[9] Tensorflow Object Detection API. online: https://github.com/tensorflow/models/tree/master/research/object_detection, Sept 2019. 5

[10] Deep Learning for Animal Detection in Man-made Environments. online: https://github.com/abhineet123/animal_detection, January 2020. 2

[11] V. Badrinarayanan, A. Kendall, and R. Cipolla. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(12):2481–2495, 2017. 5

[12] S. Beery, D. Morris, and P. Perona. The iWildCam 2019 Challenge Dataset. *arXiv preprint arXiv:1907.07617*, 2019. 1

[13] S. Beery, G. Van Horn, and P. Perona. Recognition in Terra Incognita. In V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, editors, *Computer Vision – ECCV 2018*, pages 472–489, Cham, 2018. Springer International Publishing. 1

[14] S. Beery, G. Van Horn, and P. Perona. Recognition in terra incognita. In V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, editors, *Computer Vision – ECCV 2018*, pages 472–489, Cham, 2018. Springer International Publishing. 2

[15] R. Benenson, S. Popov, and V. Ferrari. Large-scale interactive object segmentation with human annotators. *CVPR*, abs/1903.10830, 2019. 3

[16] L. Bertinetto and J. Valmadre. SiamFC - TensorFlow. online: https://github.com/torrvision/siamfc-tf, September 2019. 5

[17] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr. Fully-Convolutional Siamese Networks for Object Tracking. In G. Hua and H. Jégou, editors, *Computer Vision – ECCV 2016 Workshops*, pages 850–865, Cham, 2016. Springer International Publishing. 3

[18] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. 5

[19] A. Bréhéret. Pixel Annotation Tool. https://github.com/abreheret/PixelAnnotationTool, 2017. 3

[20] G. Chen, T. X. Han, Z. He, R. Kays, and T. Forrester. Deep convolutional neural network based species recognition for wild animal monitoring. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 858–862, Oct 2014. 2

[21] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(4):834–848, 2018. 5

[22] L.-C. Chen, Y. Zhu, and G. Papandreou. DeepLab: Deep Labelling for Semantic Image Segmentation. Github, 2017. hhttps://github.com/tensorflow/models/tree/master/research/deeplab. 5

[23] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. *arXiv:1802.02611*, 2018. 5

[24] F. Chollet. Xception: Deep learning with depthwise separable convolutions. In *CVPR*, pages 1800–1807. IEEE Computer Society, 2017. 4

[25] M. Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *NIPS*, pages 2292–2300. Curran Associates, Inc., 2013. 2

[26] J. Dai, Y. Li, K. He, and J. Sun. R-fcn: Object detection via region-based fully convolutional networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 379–387. Curran Associates, Inc., 2016. 3

[27] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009. 1, 2, 5

[28] A. Dutta, A. Gupta, and A. Zissermann. VGG image annotator (VIA). http://www.robots.ox.ac.uk/ vgg/software/via/, 2016. 3

[29] A. Dutta and A. Zisserman. The VIA annotation software for images, audio and video. In *Proceedings of the 27th ACM International Conference on Multimedia*, MM '19, New York, NY, USA, 2019. ACM. 3

[30] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010. 1

[31] C. Feichtenhofer, A. Pinz, and A. Zisserman. Detect to track and track to detect. In *ICCV*, 2017. 4

[32] D. Forslund and J. Bjarkefur. Night vision animal detection. In *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pages 737–742, June 2014. 2

[33] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013. 1

[34] R. Girshick. Fast r-cnn. In *ICCV*, pages 1440–1448, Dec 2015. 3

[35] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CVPR*, pages 580–587, 2014. 3

[36] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Region-based convolutional networks for accurate object detection and segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38:142–158, 2016. 3

[37] M. K. Grace, D. J. Smith, and R. F. Noss. Reducing the threat of wildlife-vehicle collisions during peak tourism periods using a roadside animal detection system. *Accident Analysis & Prevention*, 109:55–61, 2017. 2

[38] P. Gray. Awesome deep ecology. online: https://github.com/patrickcgray/awesome-deep-ecology, Month = August, Year = 2019, Owner = Tommy, Timestamp = 2018.02.02. 1

[39] D. Gupta. Image Segmentation Keras : Implementation of Segnet, FCN, UNet and other models in Keras. Github, 2017. https://github.com/divamgupta/image-segmentation-keras. 5

[40] W. Han, P. Khorrami, T. L. Paine, P. Ramachandran, M. Babaeizadeh, H. Shi, J. Li, S. Yan, and T. S. Huang. Seq-nms for video object detection. *CoRR*, abs/1602.08465, 2016. 4

[41] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick. Mask R-CNN. In *ICCV*, pages 2980–2988, 2017. 5

[42] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, June 2016. 4

[43] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. *CoRR*, abs/1603.05027, 2016. 4

[44] G. V. Horn. Nabirds dataset. online:http://dl.allaboutbirds.org/nabirds1, August 2019. 1

[45] G. V. Horn, O. M. Aodha, Y. Song, Y. Cui, C. Sun, A. Shepard, H. Adam, P. Perona, and S. Belongie. The inaturalist species classification and detection dataset. In *CVPR*, pages 8769–8778, June 2018. 1

[46] A. Howard, M. Sandler, G. Chu, L. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam. Searching for mobilenetv3. *CoRR*, abs/1905.02244, 2019. 4

[47] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017. 4

[48] J. Huang, V. Rathod, C. Sun, M. Zhu, A. K. Balan, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. P. Murphy. Speed/accuracy trade-offs for modern convolutional object detectors. *CVPR*, pages 3296–3297, 2017. 1

[49] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy. Speed/accuracy trade-offs for modern convolutional object detectors. In *CVPR*, 2017. 5

[50] G. Jocher. YOLOv3 in PyTorch. online: https://github.com/ultralytics/yolov3, September 2019. 5

[51] S. R. J. S. Kaiming He, Xiangyu Zhang. Deep Residual Networks with 1K Layers. online: https://github.com/KaimingHe/resnet-1k-layers, April 2016. 4

[52] K. Kang, H. Li, T. Xiao, W. Ouyang, J. Yan, X. Liu, and X. Wang. Object detection in videos with tubelet proposal networks. In *CVPR*, pages 889–897. IEEE Computer Society, 2017. 4

[53] K. Kang, H. Li, J. Yan, X. Zeng, B. Yang, T. Xiao, C. Zhang, Z. Wang, R. Wang, X. Wang, and W. Ouyang. T-cnn: Tubelets with convolutional neural networks for object detection from videos. *IEEE Transactions on Circuits and Systems for Video Technology*, pages 1–1, 2018. 4

[54] K. Kang, W. Ouyang, H. Li, and X. Wang. Object detection from video tubelets with convolutional neural networks. In *CVPR*, pages 817–825, June 2016. 4

[55] R. Kawamura. Rectlabel: An image annotation tool to label images for bounding box object detection and segmentation. online: https://rectlabel.com/, 2019. 3

[56] B. Kellenberger, D. Marcos, S. Lobry, and D. Tuia. Half a percent of labels is enough: Efficient animal detection in UAV imagery using deep cnns and active learning. *CoRR*, abs/1907.07319, 2019. 2

[57] B. Kellenberger, D. Marcos, and D. Tuia. Detecting mammals in uav images: Best practices to address a substantially imbalanced dataset with deep learning. *Remote Sensing of Environment*, 216:139 – 153, 2018. 1, 2

[58] B. Kellenberger, D. Marcos Gonzalez, and D. Tuia. Best practices to train deep models on imbalanced datasets - a case study on animal detection in aerial imagery. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 01 2018. 1, 2

[59] B. Kellenberger, M. Volpi, and D. Tuia. Fast animal detection in uav images using convolutional neural networks. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 866–869, July 2017. 2

[60] A. Khosla, N. Jayadevaprakash, B. Yao, and L. Fei-Fei. Novel dataset for fine-grained image categorization. In *First Workshop on Fine-Grained Visual Categorization, CVPR*, Colorado Springs, CO, June 2011. 1

[61] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, T. Duerig, and V. Ferrari. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *arXiv:1811.00982*, 2018. 1

[62] S. Li, J. Li, W. Lin, and H. Tang. Amur tiger re-identification in the wild. *CoRR*, abs/1906.05586, 2019. 1, 2

[63] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei. Fully convolutional instance-aware semantic segmentation. In *CVPR*, pages 4438–4446, 2017. 5

[64] T.-Y. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie. Feature pyramid networks for object detection. *CVPR*, pages 936–944, 2017. 3

[65] T.-Y. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. *ICCV*, pages 2999–3007, 2017. 2, 3

[66] 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 D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing. 1

[67] L. Liu, W. Ouyang, X. Wang, P. W. Fieguth, J. Chen, X. Liu, and M. Pietikäinen. Deep learning for generic object detection: A survey. *CoRR*, abs/1809.02165, 2018. 1

[68] M. Liu and M. Zhu. Mobile video object detection with temporally-aware feature maps. *CVPR*, pages 5686–5695, 2018. 4

[69] M. Liu, M. Zhu, M. White, Y. Li, and D. Kalenichenko. Looking fast and slow: Memory-guided mobile video object detection. *CoRR*, abs/1903.10172, 2019. 4

[70] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In B. Leibe, J. Matas, N. Sebe, and M. Welling, editors, *Computer Vision – ECCV 2016*, pages 21–37, Cham, 2016. Springer International Publishing. 3

[71] H. J. Mönck, A. Jörg, T. von Falkenhausen, J. Tanke, B. Wild, D. Dormagen, J. Piotrowski, C. Winklmayr, D. Bierbach, and T. Landgraf. Biotracker: An open-source computer vision framework for visual animal tracking. *ArXiv*, abs/1803.07985, 2018. 2

[72] M. S. Norouzzadeh, A. Nguyen, M. Kosmala, A. Swanson, M. S. Palmer, C. Packer, and J. Clune. Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning. *Proceedings of the National Academy of Sciences*, 115(25):E5716–E5725, 2018. 2

[73] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22:1345–1359, 2010. 1

[74] J. Parham, C. Stewart, J. Crall, D. Rubenstein, J. Holmberg, and T. Berger-Wolf. An animal detection pipeline for identification. In *WACV*, 2018. 1

[75] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. V. Jawahar. Cats and dogs. In *CVPR*, 2012. 1

[76] J. Patman, S. C. J. Michael, M. M. F. Lutnesky, and K. Palaniappan. Biosense: Real-time object tracking for animal movement and behavior research. In *2018 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, pages 1–8, Oct 2018. 2

[77] P. H. O. Pinheiro, R. Collobert, and P. Dollár. Learning to segment object candidates. In *NIPS*, 2015. 5

[78] P. O. Pinheiro, T. Lin, R. Collobert, and P. Dollár. Learning to refine object segments. In *ECCV*, 2016. 5

[79] J. Redmon. Darknet. online: https://github.com/pjreddie/darknet, August 2018. 5

[80] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You Only Look Once: Unified, Real-Time Object Detection. *CVPR*, pages 779–788, 2015. 3, 4

[81] J. Redmon and A. Farhadi. YOLO9000: Better, Faster, Stronger. *CVPR*, pages 6517–6525, 2017. 2, 3, 4

[82] J. Redmon and A. Farhadi. YOLOv3: An Incremental Improvement. *CoRR*, abs/1804.02767, 2018. 2, 3, 4

[83] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 91–99, Cambridge, MA, USA, 2015. MIT Press. 2, 3

[84] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, June 2017. 3

[85] N. Rey, M. Volpi, S. Joost, and D. Tuia. Detecting animals in african savanna with uavs and the crowds. *ArXiv*, abs/1709.01722, 2017. 2

[86] F. Romero-Ferrero, M. G. Bergomi, R. C. Hinz, F. J. H. Heras, and G. G. de Polavieja. idtracker.ai: tracking all individuals in small or large collectives of unmarked animals. *Nature Methods*, 16(2):179–182, 2019. 2

[87] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI (3)*, volume 9351 of *Lecture Notes in Computer Science*, pages 234–241. Springer, 2015. 5

[88] B. Russell, A. Torralba, J. Yuen, and et. al. Labelme: The open annotation tool. online: http://labelme2.csail.mit.edu/Release3.0/index.php, 2019. 3

[89] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, pages 4510–4520, 2018. 4

[90] S. Schneider, G. W. Taylor, and S. C. Kremer. Deep learning object detection methods for ecological camera trap data. *CoRR*, abs/1803.10842, 2018. 2

[91] S. Schneider, G. W. Taylor, S. Linquist, and S. C. Kremer. Past, present and future approaches using computer vision for animal reidentification from camera trap data. *Methods in Ecology and Evolution*, 10(4):461–470, April 2018. 2

[92] V. H. Sridhar, D. G. Roche, and S. Gingins. Tracktor: Image-based automated tracking of animal movement and behaviour. *Methods in Ecology and Evolution*, 10(6):815–820, 2019. 2

[93] A. Swanson, M. Kosmala, C. Lintott, R. Simpson, A. Smith, and C. Packer. Snapshot serengeti, high-frequency annotated camera trap images of 40 mammalian species in an african savanna. *Scientific Data*, 2:150026–, June 2015. 1

[94] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 4278–4284, 2017. 4

[95] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 4

[96] C. Szegedy, A. Toshev, and D. Erhan. Deep neural networks for object detection. In *NIPS*, 2013. 1, 3

[97] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, pages 2818–2826, 2016. 4

[98] M. A. Tabak, Mohammad, S. Norouzzadeh, and D. W. W. et. al. Machine learning to classify animal species in camera trap images: Applications in ecology. *Methods in Ecology and Evolution*, 10(4):585–590, 26 November 2018. 1, 2

[99] A. Ter-Sarkisov, J. D. Kelleher, B. Earley, M. Keane, and R. J. Ross. Beef cattle instance segmentation using fully convolutional neural network. In *BMVC*, 2018. 2

[100] Trieu. Darkflow. online: https://github.com/thtrieu/darkflow, March 2018. 5

[101] D. Tuia, M. Volpi, L. Copa, M. Kanevski, and J. Munoz-Mari. A survey of active learning algorithms for supervised remote sensing image classification. *IEEE Journal of Selected Topics in Signal Processing*, 5(3):606–617, June 2011. 2

[102] Tzutalin. Labelimg: A graphical image annotation tool and label object bounding boxes in images. online: https://github.com/tzutalin/labelImg, 2015. 3

[103] K. Wada. labelme: Image Polygonal Annotation with Python. https://github.com/wkentaro/labelme, 2016. 3

[104] Q. Wang. DaSiamRPN. online: https://github.com/foolwood/DaSiamRPN, September 2019. 5

[105] Q. Wang. SiamMask. online: https://github.com/foolwood/SiamMask, September 2019. 5

[106] Q. Wang, L. Zhang, L. Bertinetto, W. Hu, and P. H. Torr. Fast online object tracking and segmentation: A unifying approach. In *CVPR*, 2019. 3, 4

[107] S. Wang, W. Liu, J. Wu, L. Cao, Q. Meng, and P. J. Kennedy. Training deep neural networks on imbalanced data sets. In *International Joint Conference on Neural Networks*, 2016. 5

[108] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010. 1

[109] Z. Werkhoven, C. Rohrsen, C. Qin, B. Brembs, and B. de Bivort. Margo (massively automated real-time gui for object-tracking), a platform for high-throughput ethology. *bioRxiv*, 2019. 2

[110] D. C. Wilkins, K. M. Kockelman, and N. Jiang. Animal-vehicle collisions in texas: How to protect travelers and animals on roadways. *Accident Analysis & Prevention*, 131:157–170, 2019. 2

[111] M. Willi, R. T. Pitman, and A. W. C. et. al. Identifying animal species in camera trap images using deep learning and citizen science. *Methods in Ecology and Evolution*, 2018. 1, 2

[112] S. Xie and Z. Tu. Holistically-nested edge detection. In *ICCV*, pages 1395–1403, Dec 2015. 3

[113] W. Xue, T. Jiang, and J. Shi. Animal intrusion detection based on convolutional neural network. In *2017 17th International Symposium on Communications and Information Technologies (ISCIT)*, pages 1–5, Sep. 2017. 2

[114] F. A. R. K. J. S. S. N. A. T. B. C. Yi Zhu*, Karan Sapra*. Improving semantic segmentation via video propagation and label relaxation. In *CVPR*, June 2019. 3

[115] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *NIPS*. 2014. 1

[116] H. Yousif, J. Yuan, R. Kays, and Z. He. Fast human-animal detection from highly cluttered camera-trap images using joint background modeling and deep learning classification. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–4, May 2017. 2

[117] H. Yousif, J. Yuan, R. Kays, and Z. He. Object detection from dynamic scene using joint background modeling and fast deep learning classification. *Journal of Visual Communication and Image Representation*, 55:802–815, 2018. 2

[118] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba. Scene parsing through ade20k dataset. In *CVPR*, 2017. 5

[119] C. Zhu, T. H. Li, and G. Li. Towards automatic wild animal detection in low quality camera-trap images using two-channeled perceiving residual pyramid networks. In *ICCVW*, pages 2860–2864, Oct 2017. 2

[120] Q. Zhu, J. Ren, D. Barclay, S. McCormack, and W. Thomson. Automatic animal detection from kinect sensed images for livestock monitoring and assessment. In *IEEE International Conference on Computer and Information Technology*, pages 1154–1157, Oct 2015. 2

[121] X. Zhu, Y. Wang, J. Dai, L. Yuan, and Y. Wei. Flow-guided feature aggregation for video object detection. In *ICCV*, pages 408–417, Oct 2017. 4

[122] X. Zhu, Y. Xiong, J. Dai, L. Yuan, and Y. Wei. Deep feature flow for video recognition. In *CVPR*, pages 4141–4150, July 2017. 4

[123] Z. Zhu, Q. Wang, L. Bo, W. Wu, J. Yan, and W. Hu. Distractor-aware Siamese Networks for Visual Object Tracking. In *ECCV*, 2018. 4, 7

[124] T. T. Zin, I. Kobayashi, P. Tin, and H. Hama. A general video surveillance framework for animal behavior analysis. In *2016 Third International Conference on Computing Measurement Control and Sensor Network (CMCSN)*, pages 130–133, May 2016. 2

[125] B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. In *ICLR*, 2017. 4

[126] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. In *CVPR*, pages 8697–8710, 2018. 4