2023 IEEE AFRICON

Efficient Camera Trap Image Annotation Using YOLOv5

Yuri Njathi
Centre for Data Science and Artificial Intelligence (DSAIL)
Dedan Kimathi University of Technology
Nyeri, Kenya
yuri.njathi@interns.dkut.ac.ke

Lians Wanjiku
Centre for Data Science and Artificial Intelligence (DSAIL)
Dedan Kimathi University of Technology
Nyeri, Kenya
wanjiku.lians19@students.dkut.ac.ke

Lorna Mugambi
Centre for Data Science and Artificial Intelligence (DSAIL)
Dedan Kimathi University of Technology
Nyeri, Kenya
lorna.mugambi@dkut.ac.ke

Jason N. Kabi
Centre for Data Science and Artificial Intelligence (DSAIL)
Dedan Kimathi University of Technology
Nyeri, Kenya
jason.kabi@dkut.ac.ke

Gabriel Kiarie
Centre for Data Science and Artificial Intelligence (DSAIL)
Dedan Kimathi University of Technology
Nyeri, Kenya
gabriel.kiarie@dkut.ac.ke

Ciira wa Maina
Centre for Data Science and Artificial Intelligence (DSAIL)
Dedan Kimathi University of Technology
Nyeri, Kenya
ciira.maina@dkut.ac.ke

***Abstract*— Using camera traps to acquire wildlife images is becoming more common within conservancies. The information provided by these camera traps enhances understanding of wildlife behaviour and population patterns. The detection and counting of animals present in each of the captured images is valuable information as it can be used to guide conservation efforts. Manual annotation of these wildlife images is a tedious painful process. It is becoming more common to use tools that either use AI to annotate camera trap datasets or use AI to aid in annotation. These AI tools are usually trained on species endemic to a particular region. The ability to fine-tune such models to species endemic to one's particular region is important to save much of the time conservationists manually look through the misclassified images. In this paper, we present a case study where we used a YOLOv5 object detection model trained to detect the presence and count the number of impala and other animals from a dataset collected by researchers at the Dedan Kimathi University of Technology Conservancy. We analyze the results of the AI's performance with respect to a manually annotated dataset. The model was able to annotate 72% of the dataset at a human level of accuracy. The work here shows promise with regard to time spent labelling camera trap images by leveraging the presence of particular species to auto-annotate a majority of the dataset.**

Keywords— wildlife detection; camera traps; artificial intelligence; conservation; YOLOv5;

I. Introduction

Camera traps have become useful tools in modern wildlife conservation. They allow conservationists to monitor animals in their natural state. Conservationists can understand how animals live and survive in the wild, especially in large protected habitats. Using camera traps can generate an enormous amount of image data. This data is monitored daily, weekly or monthly to understand the ecosystem interactions in conservancies and answer important conservation questions. Surveillance cameras are used in smaller conservancies and zoos. This is mainly due to how easy it is to power surveillance cameras when animals live in a relatively small controlled area, looking through the surveillance footage is manageable. In larger conservancies, spanning hundreds or thousands of acres setting up surveillance equipment would be a harder and more expensive task. In larger conservancies, camera traps are used for image and video collection. Camera traps have the following main components:

1. A camera used to capture images and videos.
2. A power source such as batteries or solar panels.
3. A triggering device used to detect the presence of animals and activate the camera. Common triggering devices include infrared, laser or sound sensors.
4. A removable memory card is used to store the images captured by the camera traps.
5. A protective housing is used to protect the camera trap from being tampered with by animals or people.

Researchers usually deploy camera traps in a variety of ways. Camera traps can be placed in fixed locations or can be periodically moved around an area. They are placed in areas identified to have high levels of animal activity such as animal trails, water sources or feeding areas. When deploying camera traps, researchers typically use a large number of cameras in order to maximize the amount of data collected. Once camera traps are set up, researchers check them every few weeks in order to retrieve the memory cards. The memory cards are usually brought back to the lab where they are analyzed and the data is stored. The data is used to make informed decisions about conservation management such as developing strategies for protecting endangered species or identifying areas where more resources are needed. Camera trap data is used to inform policy decisions. A typical camera trap looks like the one in Figure 1 [1].

Examples of camera trap use cases are Mugambi et al [1] where 4 camera traps were used to collect 8524 images of wildlife at the Dedan Kimathi University of Technology Conservancy from June to December 2021 [1]. Another example is Palmer et al [2] where 225 camera traps were used over an area of 1125 km^2 capturing 938,596 images from July 2010 to December 2013[2].

There are 1.5 million species of lifeforms on earth [3] with different conservation statuses per the IUCN Red List of threatened species [4]. This can be used to inform conservation strategies and help prevent the occurrence of endangered species [5]. The end goal of monitoring any

2023 IEEE AFRICON | 979-8-3503-3621-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/AFRICON55910.2023.10293724

979-8-3503-3621-4/23/$31.00 ©2023 IEEE

species in an area is to understand the trend of increase or decrease that it follows. Camera traps provide a cheap solution for the temporal collection of images, which can inform conservationists on key questions over a broad span of time. With the help of AI, conservation questions can quickly and accurately be answered by the processing of the vast volumes of images taken by camera traps, especially if done over regular periods of time.

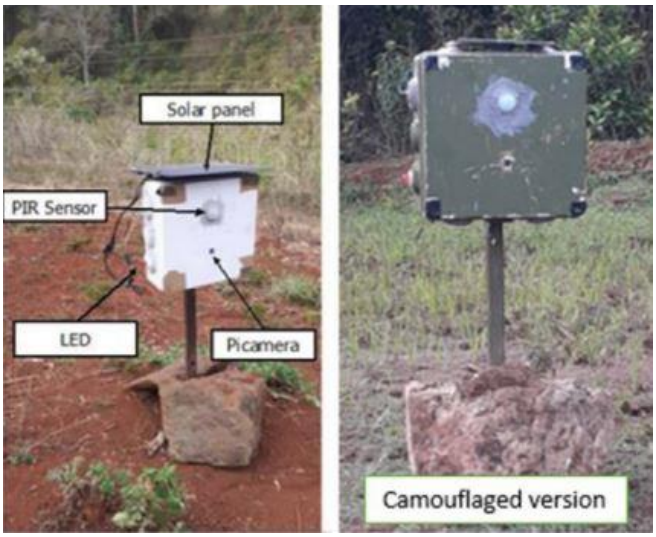

Figure 1 Camera Trap Deployed at Dedan Kimathi Conservancy [1]

II. Background and Related Work

Data: The work we embarked on centred on the DSAIL-Porini dataset [1]. This dataset consists of 8524 images from four camera traps and 6 main species of animals [1].

AI-related annotation: The use of deep learning and object detection models to annotate animal datasets has been done and is not a novel concept [6][7]. Zakria et al [6] used YOLOv3 on an animal dataset consisting of six species endemic to Ethiopia. Their dataset consisted of 3068 images with each animal being present in at least 415 images. They used augmentation, to close the gap between training and testing images. In their study, they were able to detect Ethiopian endemic animal species. A study by Norouzzadeh et al [7] used deep neural networks to annotate a dataset containing 48 species in the 3.2 million image Snapshot Serengeti dataset. The deep neural network used classification to identify, count and describe species beha vior [7]. The behaviour harnessed was movement, resting or eating. Their deep neural networks automatically identified animals with a 93.8% accuracy.

Norouzzadeh et al [7] produced a model that saved over 8 years of human labelling effort showing the gains possible when using deep learning along with camera traps. They concluded that although their deep neural network performed well on the Snapshot Serengeti dataset, performance worsened for rare classes. Their work is evidence that annotation on a large scale, millions of images, is possible. Image classification faces difficulty in counting and detecting the behaviour of species as seen in Norouzzadeh et al [7]. The application of deep neural networks in object detection may prove more useful as seen in Zakria et al [6].

Object Detection Algorithms: The three commonly used algorithms for object detection are You Only Look Once (YOLO) [8], Single Shot Detector (SSD)[9] and Region-based Convolutional Neural Network (R-CNN)[10]. YOLO uses regression to literally only looks once at an image to predict what objects are present and where they are [8]. A single CNN simultaneously predicts multiple bounding boxes and class probabilities assigned to those boxes. YOLO runs at about 45 frames per second [8][11]. YOLO divides each image into an SxS grid, with each grid predicting N boxes. From those SxSxN boxes, it classifies each box for every class and picks the highest-class probability as seen in Redmon et al [8].

Single Shot Detectors (SSD) are similar to YOLO detectors. SSDs also use a single deep neural network. Unlike YOLO, they use feature maps of each convolutional layer to predict the bounding boxes. These feature maps have different resolutions and can handle objects of various sizes. SSD is a simpler a method, it encapsulates all computation into a single network. SSDs run a 3x3 convolutional kernel on them to predict bounding boxes and classification probability. The most common SSD is RetinaNet [9][11].

R-CNNs approach pixels by classifying the pixels that make up the object in the identified bounding box. It employs a two-stage approach, using a region-based network to propose potential locations of objects, followed by a second neural network to classify and detect these objects, with a pixel mask to provide precise outlines for the objects in question [10][11].

Comparison of different object detection algorithms:

YOLO is fast and uses little processing memory but struggles where multiple objects are in a single grid or where objects are very small.

SSDs have trouble recognizing small objects [11]. SSDs match default boxes to the ground truth boxes.

R-CNN is the most accurate though it requires more resources in terms of storage and processing power for detection. R-CNN operates at a lower frame rate than YOLO and SSD. YOLO may be the better option if accuracy is not a priority or if the images have simple objects in black-and-white or on a clear background. If the images are complex and accuracy is of utmost importance, R-CNN is the better choice.

III. Objectives

Main Objective

The main objective of our work is to design a customizable system that can annotate a majority of the dataset with species, species count and species localization in the images for the animals in the dataset , thus, freeing up more time for conservationists to concentrate their efforts on more threatened species without losing track of the population numbers in less threatened species or of less interesting.

IV. Methodology

A. Data Collection and analysis

Wildlife images were obtained from the DSAIL-Porini dataset [1]. These images were collected at the Dedan Kimathi University of Technology Conservancy and a data paper by Mugambi et al [1] includes information on the species count and the motion-sensor cameras traps for data collection. The

images from the raspberry Pi 2 and Raspberry Pi Zero have an image size of 1280 x 720 pixels while images from the OpenMV Cam H7 have an image size of 640 x 480 pixels. All in JPG format. The dataset contains six main species, the impala, bushbuck, Sykes' monkey, defassa waterbuck, common warthog and Burchell's zebra.

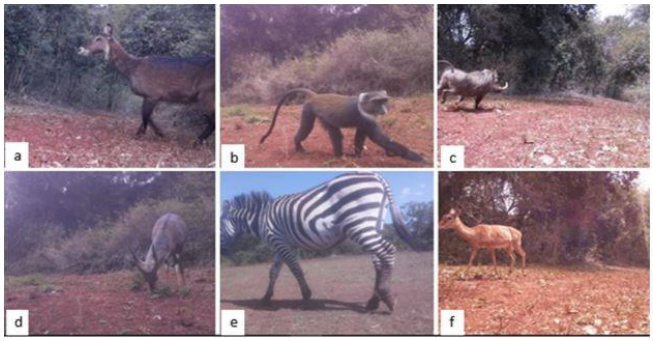

Figure 2 The six main species in the DSAIL-Porini dataset; Waterbuck, Sykes' Monkey, Warthog, Bushbuck, Zebra and Impala

Table 1 Unique species per Image

Unique species	**Number of images containing species**	**Percentage of images containing species**
Impala	5250	61.6
Monkey	48	0.6
Impala, Monkey	4	0.0
Bushbuck	312	3.7
Waterbuck	594	7.0
Warthog	1358	15.9
Zebra	422	5.0
Impala, Warthog	207	2.4
Zebra, Impala	22	0.3
Can't tell	140	1.6
Waterbuck, Impala	12	0.1
Bushbuck, can't tell	1	0.0
Impala, Zebra	154	1.8
Total	8524	100

The key takeaways from Table 1:

1. 61.6 % of all images have at least one impala. An impala detector would eliminate the need to manually annotate 61.6% of the images.
2. 87% of all the images contain Impala, Zebra and Warthog.

We trained our model on "Impala" class and "Other" class. Where "Other" class was a combination of all non-Impala species into a single class.

B. Review of YOLOv1 and YOLOv5 Detection Algorithms

We used the YOLOv5 model to understand how its layers are structured and the theory behind its performance. We look at the layers that made up the first YOLO model and look at the improvements made to have the YOLOv5 model. In Redmon et al [8], YOLO struggles with objects that appear in groups such as a flock of birds. The YOLOv1 model learns to predict bounding boxes from data, it struggles to generalize to objects in new or unusual aspect ratios or configurations. We saw this when we saw a bushbuck in a new environment, see Figure 13.

YOLO combines separate components of object detection into a single neural network. It uses features from the entire image to predict each bounding box. YOLO divides an input image into an S x S grid. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object. Each grid cell predicts B bounding boxes and confidence scores for those boxes. The YOLO model is designed as a convolutional neural network. The initial convolutional layers extract features while the fully connected layers predict the output probabilities and coordinates. It is inspired by GoogLeNet by Szegedy et al [12]. It has 24 convolutional layers followed by 2 fully connected layers. Unlike GoogLeNet which uses inception modules after its convolutional layers, YOLO uses 1 x 1 reduction layers followed by 3 x 3 convolutional layers similar to Lin et al [13]. The final output is a 7 x 7 x 30 tensor of predictions.

YOLOv5 was proposed in 2020 [14]. The core features of the YOLOv5 model are:

1. Automated data augmentation where a combination of data augmentation techniques is used to improve the accuracy of its predictions.
2. Efficient neural network design that reduce the number of parameters and improve the accuracy of its predictions
3. Cross-Stage Partial (CSP) connections to allow in information to move from one layer to another.
4. Multi-Scale training helps the model learn features at different scales this helps the model to detect objects of different sizes in an image.
5. Faster training due to the above features.

YOLOv5 improved on YOLOv1 by adding a Focus module, CBL module, CSP module, SPP module, Concat module and Upsample module, these modules refine and merge image features to overcome the problem of missed and mischeck in multi-scale feature target detection. There are 5 YOLOv5 models [14], they are YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l and YOLOv5x. Their internal structures are the same, except for the depth_multiple and width_multiple that control the depth of the models and the number of convolution cores.

We used YOLOv5s in our work whose architecture incorporates 2 kinds of CSP modules. The Leaky ReLU activation function is used in its 2 CSP modules, the first CSP structure is the backbone network while the other strengthens the integration ability of future networks. The SPP is to extract image features at a deeper level while keeping the size of the input and output unchanged.

A leaky Rectified Linear Unit is a type of activation function based on a ReLU but instead of a flat negative slope, it has a small non-zero gradient when the unit is not active. This prevents dead neurons from occurring due to zero gradients. Leaky ReLUs help speed up training, improve generalization performance and can lead to higher accuracy in deep networks.

C. Model Evaluation Metrics

To evaluate the YOLOv5 model we checked their mean Average Precision (mAP) and Intersection over Union measures. The Mean Average Precision is calculated by first calculating the Average Precision (AP) for each class in the dataset. AP is calculated by taking the area under the Precision-Recall curve for each class.

$$AP = \frac{\Delta recall}{N}\left(\sum max(precisionatrecall)\right)$$

N is the number of data points in the Precision-Recall curve

$$Precision = \frac{True\ Positives}{True\ positives + False\ positives}$$

$$Recall = \frac{True\ positives}{True\ positives + False\ negatives}$$

mAP is calculated as the average of all the class APs

$$mAP = \frac{\sum_{n=1}^{C} AP_{\mathrm{n}}}{C}$$

C is the number of classes in the dataset and AP is the Average Precision per class. An mAP of 1 is expected for an ideal model.

D. Data Pre-processing and data labelling

Before we could use the dataset for object (animal) detection, it was necessary to label a portion of the dataset according to the YOLO format using the publicly available LabelImg annotation tool the process is seen in Figure 3. In the YOLO labelling format, a text file with the same name is created for each image file in the same folder. Each text file contains annotations for the corresponding image file, including its object class, object coordinates, height and width. For each object, a new line is created as in Figure 4.

Figure 3 labelImg bounding box annotation

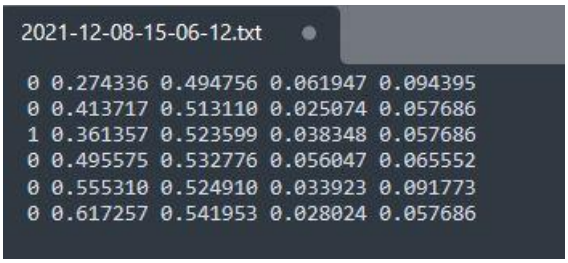

Figure 4 YOLO labelling format

E. Object Detection Model Training

For training and evaluation, 550 training images were used, 256 images had at least one impala. 176 validation images were used with 88 images having at least one impala. Some images had impalas and other species. Trained on over 500 epochs on Google Colab's Nvidia T4 GPU, with a GPU memory of 16GB and 1.59 GHz GPU memory clock taking 3 hours 20 minutes to complete training.

Table 2 Training and Validation metric scores

State	Precision	Recall	mAP
Training	0.851	0.674	0.746
Validation	0.839	0.707	0.795

F. Object Detection Model Testing and evaluation

The trained YOLOv5 model was inferenced on a NVIDIA 2GB Jetson Nano, with a 128-core Maxwell GPU. It took 1 hour, 34 minutes and 38 seconds on all 8524 images. 2582 images had no detections representing 30% of the dataset. From investigation, images that had no detections had the following characteristics:

1. Had no animals,
2. The animals were very far away and couldn't be spotted even by human eyes
3. The images were too close up
4. The images had lens flare.

These characteristics can be seen in the images in Figures 5-8. Some images had no detections but should have had at least one or two, example Figure 7.

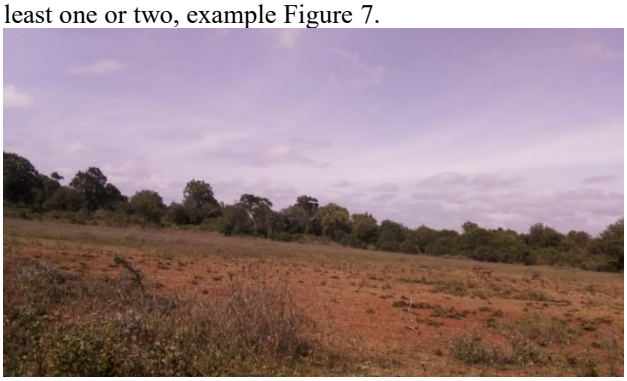

Figure 5 Seemingly-empty images with no detections

Figure 6 close up images with no detections

Figure 7 Lens Flare images with no detections

Figure 8 No detections but Impala should have been detected

V. RESULTS

A visual qualitative approach was taken on the images generated in the prediction phase. This required looking at a subset of the predicted images.

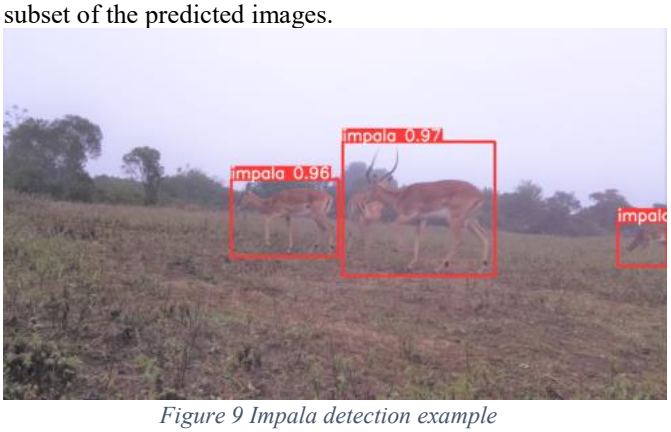

Figure 9 Impala detection example

Manual Annotation: 4 Impalas, AI annotation: 3 Impalas

Figure 10 Warthog (Other) detection example

Manual Annotation: 1 Other (Warthog), AI annotation: 1 Other

Figure 11 Multiple Warthog (Other) detections

Manual Annotation: 6 Others (Warthogs), AI annotation: 5 Others

Figure 12 Multiple Sykes' Monkey (Other) detections

Manual Annotation: 2 Others (Monkeys), AI annotation: 2 Others

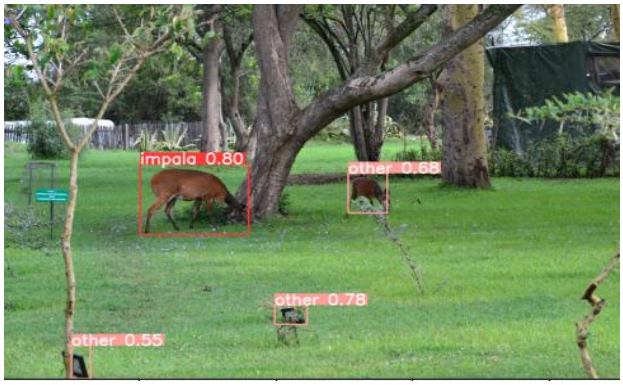

Figure 13 The YOLO model's detections in a new environment

Figure 14 Zebra detections

Manual Annotation: 2 Others (Zebras), AI annotation: 2 Others

We compared the performance of the object detector with that of a human annotator [1] and obtained a mean absolute error (MAE) of 0.9387 for all species. 72% of all 5432 images had be classified as well as a human would.

VI. Discussion

From the results, it was clearly seen that the model generalized to different species and was able to detect impala and other species in the test images as in Figure 9-14. The model took about 670ms per image. False-triggered images were accurately observed by the model. As stated YOLO struggles with generalization to new environments and new camera configurations. Figure 13 was taken by a Nikon DSLR camera in an entirely different environment with respect to the environment in the DSAIL-Porini dataset. YOLOv5 was expected to struggle with small objects but performed better than expected as seen in Figure 9-11. YOLOv5 struggled with animals that had obstructed each other, Figure 9.

Despite reviewing YOLOv5's potential disadvantages, we saw that the model performed quite well, even when trained on a small amount of data. The errors are a consequence of data collection issues mentioned by Mugambi et al [1] or YOLO's intrinsic disadvantages mentioned by Redmon et al [8].

VII. CONCLUSIONS

We were able to annotate 72% of the dataset at a human level of annotation, taking 670 ms per input image. It took 2 weeks to annotate 8524 images, taking 2 minutes and 21 seconds per input image. The model is easily customizable as we only used 550 images to achieve such results. The mean absolute error has one less animal count states that on average the AI count error is 1 individual animal. Which is relatively sound.

To improve on our work, we will train our model on the whole dataset and annotate newly obtained data from the same conservancy. There is a possible roadmap to replicating these positive results in terms of time taken and accuracy towards annotating newer wildlife camera trap images taken from the Kenyan conservancy and providing a good proof of concept for future work with other conservancies on camera trap imagery. We plan on adding all the classes to the train dataset.

Acknowledgment

We would like to thank Data Science Africa for support through the Affiliated Centre Program and NVIDIA Corporation for a hardware grant to the Centre for Data Science and Artificial Intelligence (DSAIL).

References

[1] Mugambi, L., Kabi, J. N., Kiarie, G., & Maina, C. wa. (2023). DSAIL-Porini: Annotated camera trap image data of wildlife species from a conservancy in Kenya. *Data in Brief*, *46*, 108863. https://doi.org/10.1016/J.DIB.2022.108863

[2] Palmer MS, Packer C (2018) Giraffe bed and breakfast: Camera traps reveal Tanzanian yellow-billed oxpeckers roosting on their large mammalian hosts. Afr J Ecol.

[3] About Species | WWF. (n.d.). Retrieved February 8,2023, from https://wwf.panda.org/discover/our_focus/wildlife_practice/about_species/

[4] Mace, G. M., Collar, N. J., Gaston, K. J., Hilton-Taylor, C., Akçakaya, H. R., Leader-Williams, N., Milner-Gulland, E. J., & Stuart, S. N. (2008). Quantification of Extinction Risk: IUCN's System for Classifying Threatened Species. *Conservation Biology*, *22*(6), 1424–1442. https://doi.org/10.1111/j.1523-1739.2008.01044.x

[5] IUCN Red List: https://www.iucnredlist.org/about/influence-on-conservation

[6] Endris, A., Zakria, Deng, J., Ahmednasir, M., Mohammed, J., & Kewyu, N. (2021). Towards automatic ethiopian endemic animals detection on android using deep learning. *2021 4th International Conference on Pattern Recognition and Artificial Intelligence, PRAI 2021*, 463–468. https://doi.org/10.1109/PRAI53619.2021.9550798

[7] Norouzzadeh, M. S., Nguyen, A., Kosmala, M., Swanson, A., Palmer, M. S., Packer, C., & Clune, J. (2018). Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning. *Proceedings of the National Academy of Sciences of the United States of America*, *115*(25), E5716–E5725. https://doi.org/10.1073/PNAS.1719367115/SUPPL_FILE/PNAS.1719367115.SAPP.PDF

[8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., vol. 2016-Decem, pp. 779–788, 2016, doi: 10.1109/CVPR.2016.91

[9] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, G. Cheng-Yang & A. Berg, "SSD: Single Shot MultiBox Detector" In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds) Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science(), vol 9905. Springer, Cham. https://doi.org/10.1007/978-3-319-46448-0_2

[10] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on, pages 580–587. IEEE, 2014.

[11] H. Devanathan, "The Basics of Object Detection: Yolo, SSD, R-CNN," *Medium*, 11-Oct-2022. [Online]. Available: https://towardsdatascience.com/the-basics-of-object-detection-yolo-ssd-r-cnn-6def60f51c0b. [Accessed: 22-Feb-2023].

[12] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. CoRR, abs/1409.4842, 2014.

[13] M. Lin, Q. Chen, and S. Yan. Network in network. CoRR, abs/1312.4400, 2013

[14] L. Ting, Z. Baijun, Z. Yongsheng and Y. Shun, "Ship Detection Algorithm based on Improved YOLO V5," 2021 6th International Conference on Automation, Control and Robotics Engineering (CACRE), Dalian, China, 2021, pp. 483-487, doi: 10.1109/CACRE52464.2021.9501331.