Analysis of Camera Trap Footage Through Subject Recognition

Nirnayak Bhardwaj

Analysis of Camera Trap Footage Through Subject Recognition

A Thesis

Presented to

Dr. Robert Chun

The Faculty of Department of

Computer Science

San José State University

In Partial Fulfillment

Of the Requirements for the

MSCS

By

Nirnayak Bhardwaj

December, 2021

The Designated Thesis Committee Approves the Thesis Titled

Analysis of Camera Trap Footage Through Subject Recognition

by

Nirnayak Bhardwaj

APPROVED FOR THE DEPARTMENTS OF COMPUTER SCIENCE

SAN JOSE STATE UNIVERSITY

DECEMBER 2021

ABSTRACT

Motion-sensitive cameras, otherwise known as camera traps, have become increasingly popular amongst ecologists for studying wildlife. These cameras allow scientists to remotely observe animals through an inexpensive and non-invasive approach. Due to the lenient nature of motion cameras, studies involving them often generate excessive amounts of footage with many photographs not containing any animal subjects. Thus, there is a need for a system that is capable of analyzing camera trap footage to determine if a picture holds value for researchers. While research into automated image recognition is well documented, it has had limited applications in the field of ecology. This thesis will investigate previous approaches used for analyzing camera trap footage. Studies involving traditional computer vision and machine learning techniques are reviewed. Furthermore, the datasets and additional feature recognition utilized by the techniques will be explored to showcase the advantages and disadvantages of each process, and to determine if it is possible to improve upon them.

***Keywords* – Image Classification, Machine Learning, Convolutional Neural Networks, Detectron2, Computer Vision, Camera Traps**

TABLE OF CONTENTS

I. INTRODUCTION

The advent of motion-sensitive cameras has shown that they are an integral tool for observing animals and their behavior. Ecologists are able to utilize the latest technological advancements in cameras to capture imagery of animals through an inexpensive and non-invasive approach, as well as allowing for remote observation [1]. The results of camera trap studies have been used to study biodiversity, population structure, foraging behavior, as well as providing other key information on species [2]. Due to the nature of these cameras, they're designed to operate on a frequent basis by taking a photograph whenever they detect movement. This often results in a situation where the camera captures a "false positive" due to perceived motion, but with no actual animal in the frame. Depending on the scale of a camera trap study, ecological surveys using camera traps can generate thousands of photographs, with each requiring manual processing to ensure there is an animal in the frame [3]. This results in a very time-consuming and expensive task, and is further exacerbated due to the increasing amount of camera trap studies being published [4].

Scientists have researched methods of automated image analysis and have applied them to different biological areas such as predicting gene ontology or cancer cell classification but investigations towards applying image analysis on camera trap footage are limited [18]. The typical methods of image recognition involve either traditional computer vision, where a static algorithm performs a set analysis on an image, or machine learning that involves an algorithm actively learning from its own results as it analyzes images. Several studies have researched applying such techniques on camera trap images, with methods ranging from pixel analysis to applying deep learning on image databases.

This thesis aims to explore the different image analysis methods used on camera trap footage to determine the following: How is camera trap footage analyzed? What are the differences between computer vision and machine learning techniques? How can either computer vision or machine learning be applied to automate analysis of camera trap footage? This thesis will reference journals, published papers, and technical reports. Fig 1. displays the content that this literature review will follow. The first section of the project will briefly discuss the history and purpose of camera trap footage. The second section will investigate the different image analysis techniques to determine their respective strengths and weaknesses to determine if either has a particular advantage over the other for analyzing camera traps.

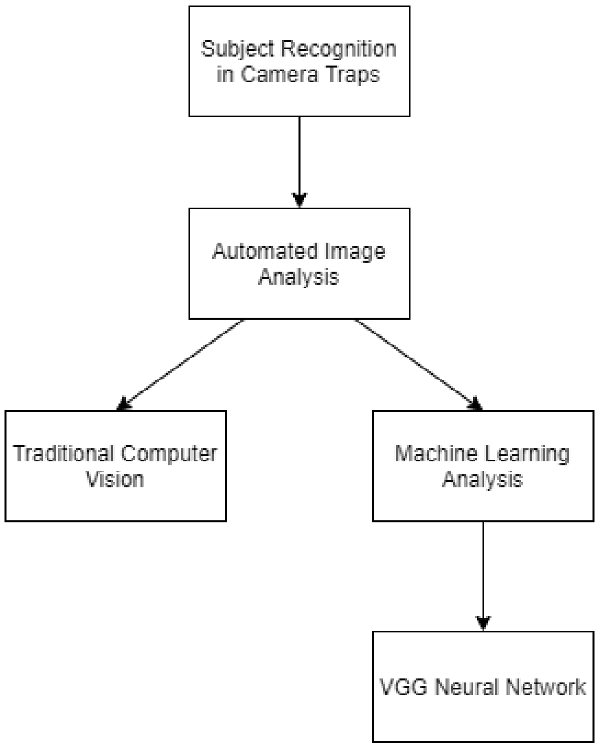

Subject Recognition in Camera Traps
Automated Image Analysis
Traditional Computer Vision
Machine Learning Analysis
VGG Neural Network

Fig. 1. Structure diagram of topics

II. BACKGROUND ON CAMERA TRAPS

The term *camera trap* is a catch-all term for remote camera systems that are designed to automatically capture photos whenever a sensor is triggered. Sensors previously utilized physical switches such as tripwires or pressure plates in order to activate, but modern designs typically have a type of infrared beam to detect motion [5]. Scientists have been using camera trap systems for decades thanks to their ability to provide affordable and accurate data while also maintaining minimum interaction between themselves and the animal subjects. Due to this, it can be argued that camera traps are superior to human surveillance techniques [14]. The data that is obtained from camera traps can provide valuable information such as: Which animal species are present in an area, how many individuals are present, when is a certain species most active, what food items are consumed by certain animals, biodiversity, habitat use, and so on [11]. Hence, there is a need for a model that can analyze information provided from camera trap data.

III. AUTOMATED IMAGE ANALYSIS

Since camera traps have demonstrated their effectiveness in the field, an increasing number of ecological studies have utilized them as the main tool for observation. It's no longer uncommon to see biological surveys that rely solely on camera traps to obtain their data. The data can contain key information about the different species present at a location, population counts, behavior patterns, and dietary habits. However, studies that use camera traps to such a degree are large scale operations in both time and size. [4] details an extensive camera trap study from New Mexico that utilized two dozen camera traps and captured over 110,000 images over a six-month time period. Scientists spent the next year manually analyzing and labeling photographs to determine which photos contain useful information on the animals they were surveying. In addition, the job of manually processing images introduces the possibility of human errors due to the tedious nature of the task [5].

Similarly, [6] was also based on a similar vein and details the efforts from another well-known camera trap study, the Snapshot Serengeti camera-trap project based in Tanzania, Africa. The SS project is similar to the New Mexico study in the fact that researchers would also manually analyze each of the photographs from the traps. However, the scale of the study was significantly larger and consisted of millions of photographs that had been taken from hundreds of camera traps. In order to undertake such a task, the project had the support of thousands of "citizen scientists", who are ordinary web users providing their own analysis on photos captured by the cameras. The labeling of images took years to complete and is still currently active to this day.

Both papers conclude that manually processing the photos is too time consuming and tedious of a task and shift their focus towards systems that can automatically analyze photographs. Automated image analysis refers to the extraction of attributes from images by computers through image processing algorithms. It has been successfully applied to industries such as astronomy, allowing astrologers to extract details from pictures containing thousands of stars [7], but applications of image analysis for animal camera trap data is limited to a select few studies. Regardless, studies made so far have utilized two separate methodologies: traditional computer vision and machine learning approaches.

A. TRADITIONAL COMPUTER VISION (NON-MACHINE LEARNING)

Traditional computer vision techniques are capable of analyzing images through a consistent approach that does not require itself to learn or update its methods for extracting features. Computer vision techniques were once the predominant method for performing object recognition but have taken a backseat to machine learning in recent years. However, this is not to say that classical techniques have become obsolete. On the contrary, they have some noteworthy advantages over their machine learning counterparts. The main advantage being that computer vision typically does not require the heavy training and algorithmic costs of machine learning algorithms, which often have a significant number of parameters with an equally large number of examples to be trained for. Having a limited training dataset can often result in poor performance since a machine learning technique may be overfitting for the training data and might not adapt properly for the primary task. On the other hand, computer vision techniques do not require any prior training on datasets before they can be used for subject recognition. This allows computer vision to not be constrained by outside factors when performing a job on the main dataset. In a way, computer vison can be considered as a more direct approach to solving the problem of identification. In fact, computer vision can solve certain problems more efficiently and use less lines of code than a machine learning approach [19].

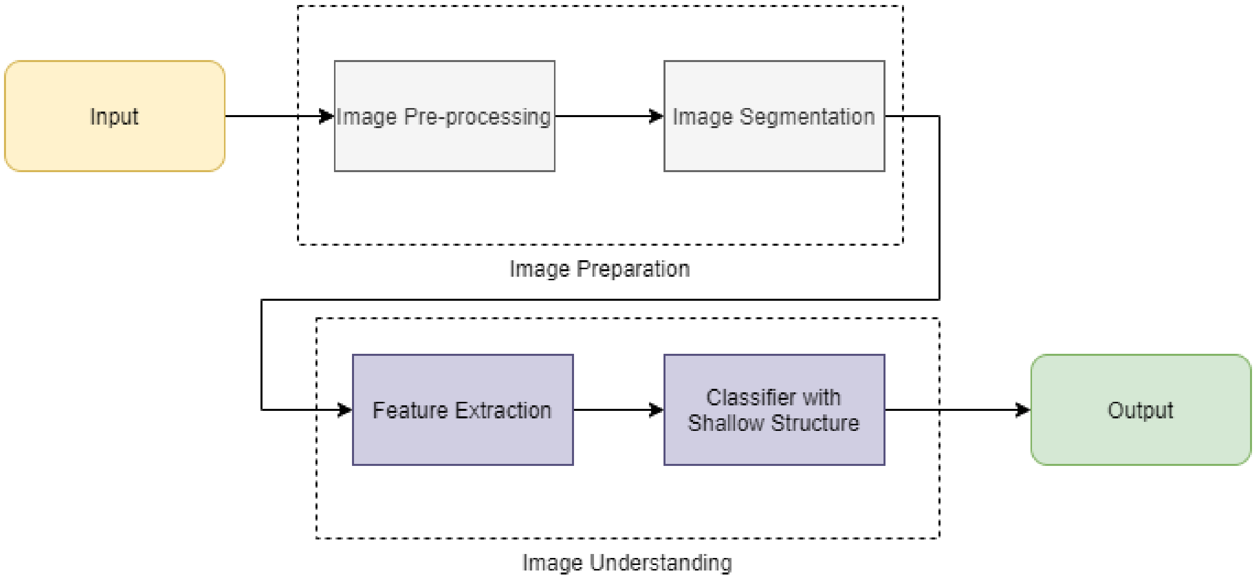


Fig. 2. Traditional Computer Vision workflow

Before the recent proliferation of machine learning techniques, several researchers attempted to apply traditional computer vision on camera trap footage. [2] performed their research on footage of Eurasian beavers using two separate methods in order to determine whether or not a photograph contained a beaver subject. The two approaches analyzed key graphical features before applying filters to compare the frame variations of images to another reference frame. The first approach compared the frame variations between the current frame and average frame instance, while the second approach compared the current frame to its subsequent frame instances to calculate the amount of pixel variation. The team applied both filter methods onto a dataset containing 1991 recordings over a 405-night period, both methods aimed to measure the changes in pixel values within a recording. At the end of the study, it was determined that filter 2 performed slightly better than filter 1, thanks to higher detection rates in a variety of environmental conditions. Filter 2 was able to achieve a detection rate of 53% when the false positive rate (the number of misidentified recordings divided by the total image set count) was 5%. When the false positive rate was allowed to be 20%, the detection rate increased to 76%. Hence, demonstrating that the model could discern animals to a reasonable but not an outstanding degree. The team also noticed several shortcomings in their model, the main issue being that the model was only capable of recognizing beavers and could not distinguish other species. Changing the model to recognize more than one species was not feasible and would take a significant amount of time and effort as described by the team. Another issue mentioned was the potential of the subject remaining stationary during the recordings, resulting in the changing pixel values being too low and the footage would be deemed as insignificant. Lastly, the different environmental conditions encountered by the cameras, such as night/day and dry/wet conditions, were enough to cause significant variation in recorded observations.

In a similar vein to the aforementioned paper, [5] also utilized computer vision techniques on camera trap footage of wild Ocelots. Likewise, the team decided on two methods that similarly compared subsequent images based on the notion that if a camera trap was triggered, then two photographs taken within a very small-time frame would only differ by the absence of an animal in one of the photographs. Method 1 compared the difference between two images pixel by pixel, while method 2 utilized a squared window method that took the "difference" of two images using a squared window of size *m*. As the team predicted the pixel by pixel struggled to obtain meaningful results since it was too sensitive to changes in projected shadows and leaves scattered by the wind. The square method technique was more robust and successful in determining which image contained an animal. But the authors do not mention the percentage of success, nor do they mention the size of the dataset tested upon, resulting in a conclusion that leaves more to be desired. Like the study performed on beavers, this study was only capable of focusing on one animal species, as the team mentioned that recognizing more than one animal species would be too tedious. The impression given by computer vision techniques demonstrates a lackluster ability to perform accurate subject recognition on camera trap data, as well as not being adaptable enough to perform recognition on new subjects without making drastic changes to the existing model.

B. MACHINE LEARNING ANALYSIS

Machine learning refers to a group of algorithms that can analyze and interpret data, learn from it, and depending on what was learned make the best possible choice; essentially allowing computers to solve tasks without explicitly being told how they should solve them. There are two types of machine learning methods, supervised and unsupervised learning. Supervised learning introduces example inputs along with their correct outputs, the goal is to learn a consistent format for mapping inputs to outputs. The labeling helps to serve the model by allowing itself to measure its accuracy and learn over time. Unsupervised learning, on the other hand, is provided unlabeled data and works on its own to discover inherent structures within the data. The end goal for unsupervised learning is to understand new insights and discover hidden features from large amounts of data, rather than being able to accurately predict outcomes of new data.

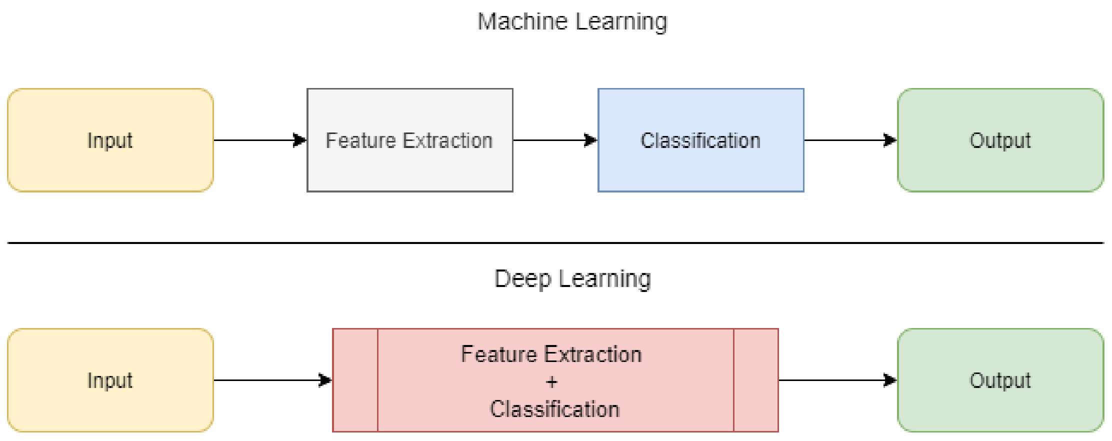


Fig. 3. Machine Learning compared to Deep Learning

For image analysis, studies typically utilize supervised learning with a computational model known as deep learning, a subset of machine learning based on artificial neural networks inspired by how biological systems learn from past experiences [3]. Similar to the human brain, it consists of many computing cells or 'neurons' that run simple tasks while working with other cells in order to reach a decision. In traditional machine learning, models can often succumb to environmental changes, but deep learning models can adapt to these fluctuations by constant feedback. When compared to classical computer vision methods, deep learning models are trained rather than explicitly programmed. This quirk allows for greater flexibility in real-world applications since the models can be retrained against any specific dataset while computer vision models are typically focused on a single domain. Subject recognition through deep learning has revolutionized the way scientists use computers to recognize key subjects in images and videos. It has already proven itself successful in multiple fields including biology; however, it has only seen a handful of applications in camera trap studies [16]. It's ability for self-improvement may be limited for such footage due to the large degrees of variation between each image. Factors such as weather, time of day, shadows, and the number of animals all acted as potential obstacles for traditional computer vision techniques, so deep learning algorithms must be able to account for these impediments [8].

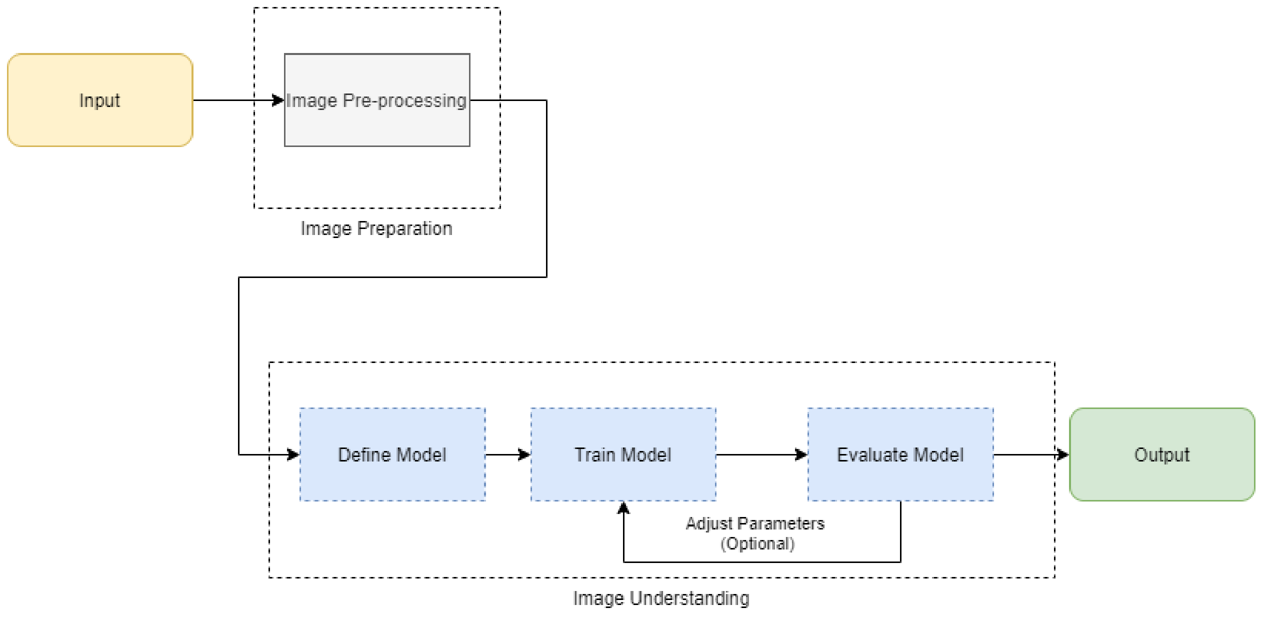


Fig. 4. Deep Learning workflow

Interested in the possible benefits of deep learning models, recent studies have attempted to measure its effectiveness on camera trap footage. [6] was interested in analyzing footage of the different animal species found in a national park of Mozambique, Africa. The team trained a convolutional neural network (CNN), a type of feedforward deep neural network with multiple layers where each “neuron” uses convolutional operations to extract information from specific overlapping sections of the earlier layers. Through the recent boom in modern machine learning, CNNs have proven themselves to be incredibly effective in extracting various features from image data and are now considered the industry standard. The CNN was applied on a previously untested and fully annotated dataset of the top 20 most common species in the park. The dataset contained 111,467 images split into three subsets consisting of 85% training, 5% validation, and 10% testing. The reason why the team confined their selection to the top 20 species was to avoid rarer animals that might have degraded the network’s performance. A lone deep learning algorithm ‘VGG-16’ was the main component selected to train the network. Next, Guided Grad-CAM (GG-CAM) methods were used in the last layer of the trained model to focus on specific local features from images and provide context to how the algorithm was recognizing objects. Afterwards a Mutual Information (MI) method was used in the final layer to compare common within-species features. This method allowed the network to distinguish key features such as an elephant’s trunk, or specific-colored stripes on deer. Lastly, hierarchical clustering was performed on CNN feature vectors for 6000 randomly selected images in the last fully-connected layer. The purpose of clustering was to gain an insight into how the CNN was quantifying similarities between the 20 animal species.

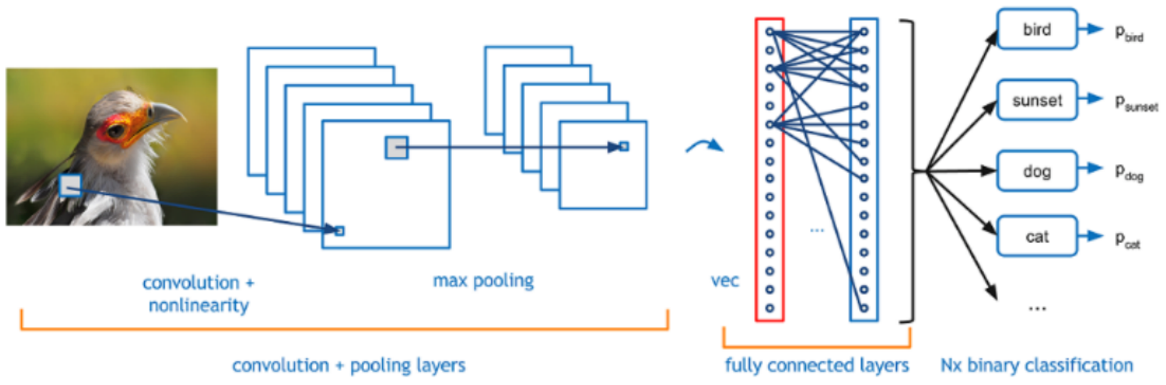
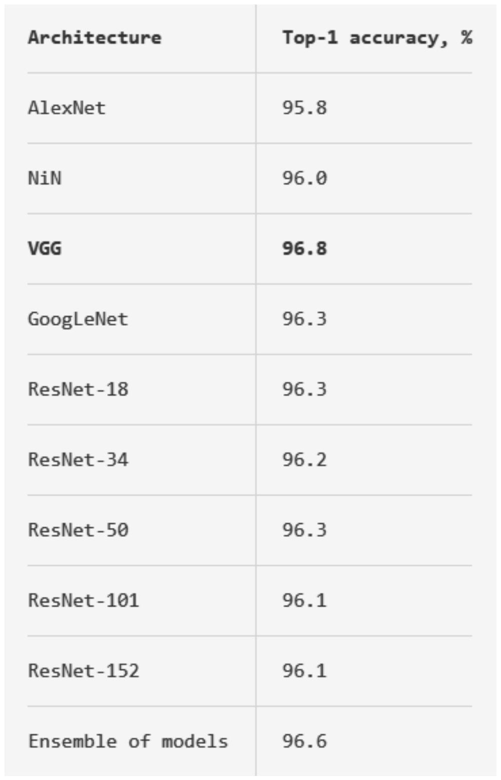


Fig 5. Building blocks of a CNN

Once the CNN was trained, the overall accuracy was 87.5%, while accuracy for all 20 species was 83.0%, the highest individual species-accuracy value was 95.2% and the lowest was 54.3%. For comparison's sake, the team trained the CNN with another deep learning algorithm, ResNet-50, which had more layers but required less parameters than VGG-16. Performance between the two algorithms provided similar results, with GG-CAM and MI results being similar while hierarchical clustering provided slightly differing results. However, no specific values for accuracy are provided. In addition, the team extracted information from the GG-CAM, MI, and clustering phases to learn of the strengths and weaknesses displayed by the CNN. They discovered that the CNN used similar techniques as humans do for recognition. One instance of this is seen for Baboons where the CNN focused on the faces and tails for the easiest identification. It was also noted that the CNN struggled when distinct features were hidden or multiple species were present in the same image.

In a similar focus, [3] also aimed to incorporate deep learning but had more ambitious goals in mind. The team noted that the majority of previous work done with camera trap footage was based on either small datasets or hand-designed detection features to find subjects. As a result, the team implemented a deep CNN and aimed to find the following information: Determine if an image contained an animal or not, identify which species is present, count the number of subjects in an image, and describe additional animal features (what behavior is being displayed or whether young are present). A large amount of data was needed for such a complex analysis and the team utilized a dataset of images from the Snapshot Serengeti (SS) project, the world's largest camera trap study. Unlike the previous study's dataset which only contained images of the top 20 most common species for optimal performance, the SS dataset contained all possible capture events recorded, meaning images having rare animals or those with no animals were also included. From previous experience, the team knew the majority of images would not contain a subject and designed a two-stage pipeline that first determined if an animal was present in the image, while the second stage extracted the key features. The first stage performed well reducing the dataset down to 301,400 images, a reduction of 75%. The dataset was further split into a training set of 284,000, an expert labeled set of 3800, and a volunteer labeled set of 1400 images respectively. The second stage focused on information extraction for the three remaining tasks and employed a single model using multitask learning. Since all the three remaining tasks were related, weights and common features could be shared amongst the tasks for better performance, requiring fewer parameters which leads to an efficient model. Lastly, the team was interested in testing multiple deep learning algorithms to discover which provided the best result. Thus, ten separate classification algorithms were selected and tested. All models exceeded 90% accuracy but the VGG model achieved the best performance with a 96.8% detection rate when determining if an animal was present [Fig. 6].



Architecture	**Top-1 accuracy, %**
AlexNet	95.8
NiN	96.0
VGG	**96.8**
GoogLeNet	96.3
ResNet-18	96.3
ResNet-34	96.2
ResNet-50	96.3
ResNet-101	96.1
ResNet-152	96.1
Ensemble of models	96.6

Fig 6. Classification algorithm performance

In regards to the other tasks, they achieved a 93.8% accuracy rate in identifying the correct species, and an 83.6% accuracy when counting the number of animals from the same species in one photo. However, their model suffered when multiple animal species was present in the same photo, an issue shared by [6]. The capability of the two-stage pipeline is apparent, as it’s able to extract far more detailed features from photos as well as provide context to the reviewers. An example of this is when the algorithm determined, by itself, that certain species of impala consistently had a black spot on their lower abdomen, a tactic that human researchers did not think to utilize when manually classifying images. The models were fortunately able to deal with the variations in environmental factors and displayed high rates of accurate identification, an issue that was not overcome by classical computer vision. Furthermore, the ability to provide additional recognition statistics to the scientists demonstrates a clear advantage towards machine learning when compared to traditional computer vision, as both the algorithm and scientists are able to learn from past experiences to adjust their techniques for better analysis.

IV. VGG NEURAL NETWORK

From the results of traditional computer vision [5] [2] and machine learning studies [3] [6], it's clear that the latter displays robust capability in analyzing camera trap footage. While the machine learning studies operated on different datasets and had different end-goals in mind, they both had large datasets that employed CNNs and utilized VGG, a famous convolutional neural network model, to train both CNNs. VGG was originally released in 2014 and was immediately heralded as one of the most capable models available for image analysis. Trained for weeks through Nvidia Titan GPUs, it made improvements over the, at-the-time industry standard, 'AlexNet'. While VGG was based on AlexNet, the designers made some alterations to the model, one was reducing AlexNet's large receptive fields of 11x11 to a much smaller 3x3 receptive field.

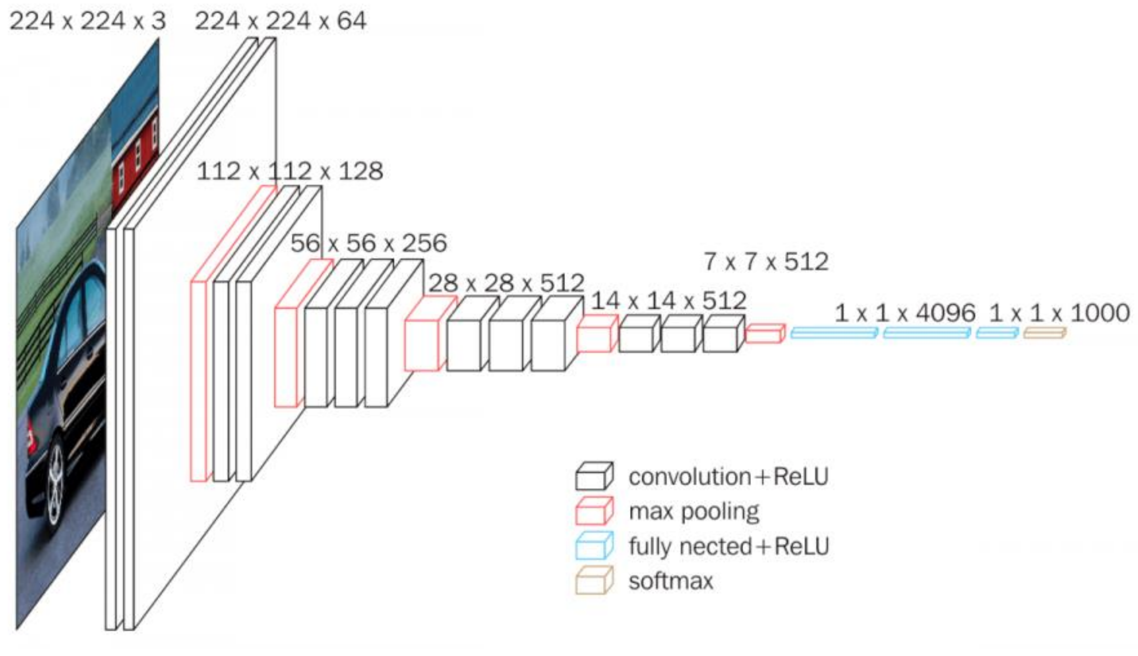


Fig 7. Architecture of VGG model

In addition, VGG has three rectified liner units (ReLU), the most commonly used type of activation function in neural networks, compared to just one unit in AlexNet allowing for a more discriminative decision function. AlexNet also has more parameters per channel at 49, while VGG has 27. These changes allow VGG to have more weight layers which provides greater performance. VGG is capable of achieving a 92% top-5 accuracy rating against ImageNet, a database consisting of over 14 million images and 1000 separate classes [21], and outperformed nine other classification algorithms in the examined study [3]. However, VGG does have one noticeable downside in the fact that it takes an extremely long time to train. The original research team trained the model for weeks using expensive Nvidia GPUs. VGG also requires a heavy architecture network due to the depth and quantity of nodes and can be difficult to use on smaller architectures.

V. HYPOTHESIS

As more and more ecologists rely on camera traps for their studies, the number of images recorded will also increase [12]. Machine learning techniques demonstrate significant capabilities over traditional computer vision techniques by accurately extracting and quantifying data with a high rate of success. The accuracy of subject recognition is noticeably higher on larger datasets, and the importance of additional feature extraction, such as counting the quantity of a species in a single photo, are both equally valuable advantages for machine learning.

However, even though the machine learning models examined earlier demonstrated high accuracy rates, they both still struggled in particular edge cases. A noticeable case was the situation where multiple animal species appeared in a single image. In this situation, both models were uncertain on how to classify the image due to conflicting classes. In the wild, it’s common for multiple species to cohabitate with one another for protection or symbiotic relationships. Thus, this is an important case to cover and can even provide insights into new inter-species relationships.

Through advanced deep learning techniques, it’s possible to create models that can perform recognition and correctly classify two classes in the same photograph [22]. This study will aim to analyze camera trap data from the Snapshot Serengeti project in order to classify images of both single animals and those containing multiple species. Machine learning models such as regional convolutional neural networks will be utilized and results will be compared to existing methods for evaluation purposes.

VI. DATASET

The dataset of images used for this project are from the Snapshot Serengeti (SS) project [24]. As mentioned earlier, the SS project has been operating in the Serengeti National Park of Tanzania for the past 12 years and has recorded millions of high-frequency annotated camera trap images for over 40 mammalian species. The reason for choosing this dataset is to compare results with the previous studies [3] [6] that used camera trap footage from African national parks. In particular, [3] also used footage from the SS project. Images from the dataset are sufficiently detailed at 2048 x 1536 resolution while following RGB color channels. Since using all the images in the dataset was out of the question, a select number of species, specifically: Warthogs, Thomson's Gazelle, Wildebeest, and Zebras, were selected to be used for the study. These animals are all herbivore species that are commonly found in the park. Additionally, there were many instances for each species where they were found solitary or grouped with the other species selected for the study, making them prime candidates for multi-subject recognition.

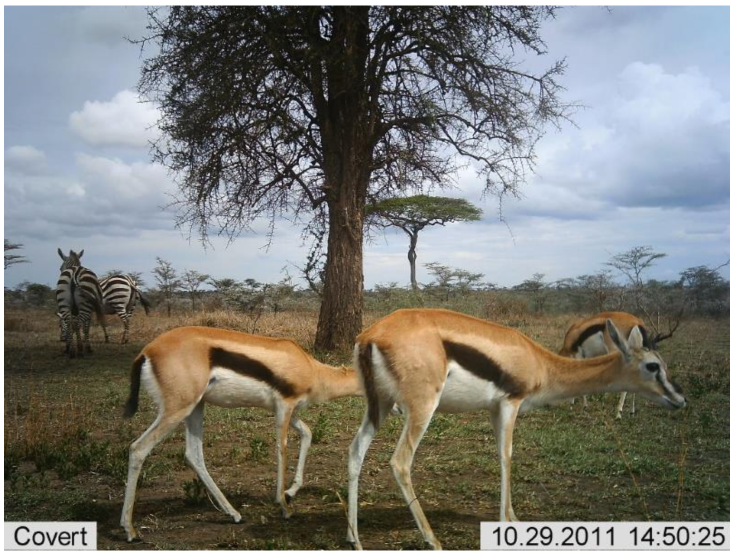


Fig 8. Example of "Thomson's Gazelle and Zebra" image

Each species had 560 camera trap photos consisting of a mixture of random images. Resulting in 2240 images with each containing a single species. In addition, another subset of images that contained a combination of “Wildebeest and Zebra” and “Thomson’s Gazelles and Zebras” were also included. They too contained 560 images respectively. The dataset had a wide variety of images showcasing animals under different conditions, times, locations, and poses, allowing for a great deal of variation between each image. Several species even had camouflage patterns for hiding in specific environments. All of these characteristics act as impeding factors for the model, making classification an arduous task.

No.	Animal Species	Training	Test	Total
1	Thomson’s Gazelle	500	60	560
2	Warthog	500	60	560
3	Wildebeest	500	60	560
4	Zebra	500	60	560
5	Thomson’s Gazelle and Zebras	500	60	560
6	Wildebeest and Zebras	500	60	560

Fig 9. Dataset Spread

VII. EVALUATION METRICS AND MODELS

A. Classification Accuracy

Classification accuracy is one of the main metrics when evaluating classification models. In simple terms, it is the ratio of the number of correct predictions over the total number of predictions made by the model.

$$Accuracy = \frac{Number\ of\ Correct\ predictions}{Total\ number\ of\ predictions\ made}$$

Fig 10. Equation for Determining Accuracy

This metric requires an equal number of samples belonging to each class. Hence, this is the reason why all classes being tested have the same number of images between them.

B. Loss

Loss is used to optimize the performance of models by penalizing incorrect classifications. It's calculated from the training and validation datasets and quantifies the performance of the model. It can also be described as the summation of errors that were encountered during those two sets. Having minimal loss through each epoch is the optimal goal, as the optimization should improve over time. There are different types of loss functions that are used for different purposes, but for this study, the chosen loss function is cross-entropy or log loss.

$$-\sum_{c=1}^{M} y_{o,c} \log(p_{o,c})$$

Fig 11. Cross-Entropy loss model.

Like most loss models, cross-entropy harshly penalizes the model for being over-confident and generating incorrect predictions, as well as correct and less confident predictions to a lesser degree.

C. Convolutional Neural Network with ResNet-18

In order to test the accuracy of the hypothesized model, a typical CNN will act as the control to compare performance metrics. In this case, a CNN using ResNet-18 will be employed on the dataset. The reasoning for choosing ResNet-18 over other classification models is due to its faster performance. It is able to maintain its high accuracy while utilizing fewer parameters than other models. Having fewer parameters reduces the space needed to store the network and shortens the overall runtime. It is also less taxing for the machine, which is an important consideration when your machine has memory limitations or an older GPU. Compared to other models such as VGG, which employs roughly 138 million parameters, ResNet is more lightweight and employs just 25.5 million parameters.

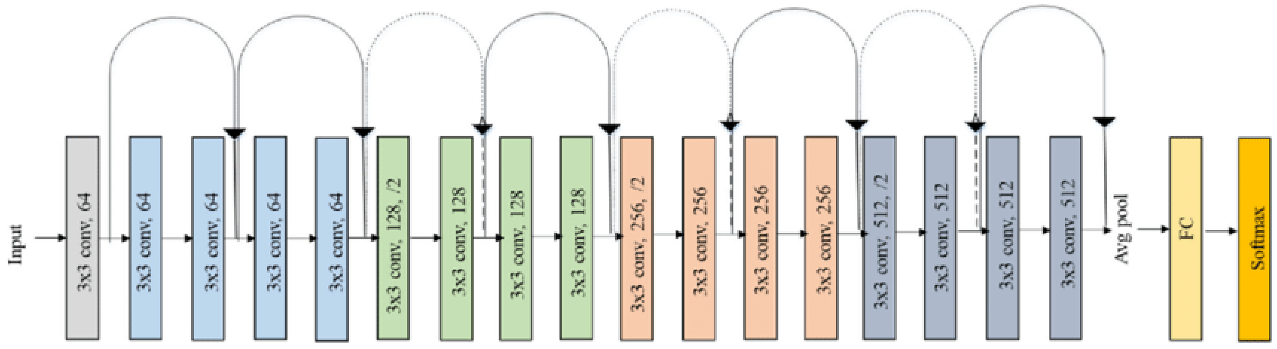


Fig 12. Layers of ResNet-18

A few initial test-runs with VGG were performed on subsets of the main dataset to determine the performance disparity with ResNet-18. Results indicated that the difference between the two on this dataset was negligible with overall accuracy differing between $\pm$.02 or 2 percent, making the ResNet-18 sufficient for this study.

D. Region Based Convolutional Neural Network with Detectron2

The main aim of this study is to test the dataset with a Region-Based Convolutional Neural Network (R-CNN). Whereas traditional CNNs perform image classification by taking an input image and outputting a class label, R-CNNs aim to determine the location of each object/class in an image with a bounding box. In real-world scenarios, it's common to encounter images that contain multiple objects, and the single class label output used by CNNs can prove troublesome. If an image contains more than one subject, the CNN would have to specify that both subjects are present and assign a unique label referencing the pair. This can be challenging for the model as it introduces the possibility of missing a particular subject, where the system may recognize just one subject and immediately apply the class label. Furthermore, the creation of unique class labels can prove troublesome from a usability aspect. If a real-world camera trap study focused on the same four species that are examined in this paper [gazelle, wildebeest, warthogs, and zebra] there could be up to 16 class labels to account for all possible combinations. This number would grow exponentially if more species are added to the study. Additional classes caused by these combinations further exacerbate classification issues and will lower overall accuracy. R-CNNs don't suffer from this issue since they focus on the individual class labels and don't need to consider class combinations [26], so the model would only need to consider the 4 classes instead of 24 possible combinations.

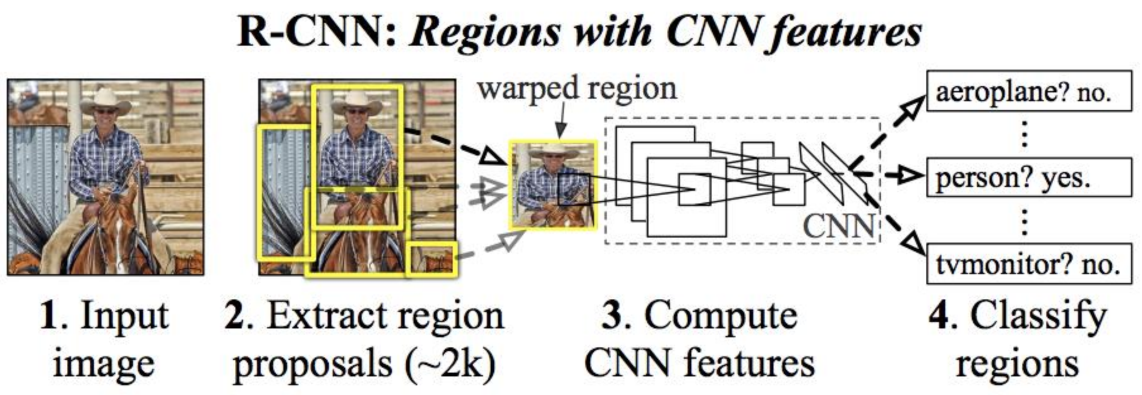


Fig 13. R-CNN Classification Process

Enter Detectron2, a flexible PyTorch based library designed as an object detection framework. Created by Facebook, it includes high-quality implementations of advanced object detection algorithms and supports cascade R-CNN, panoptic segmentation, DeepLab, and etc. [29]. Seeing as R-CNNs are well designed for tackling issues commonly encountered in camera trap studies, Detectron2 seems like an appropriate choice for applying a R-CNN model. By utilizing a pre-trained model, it's possible to fine-tune the model by training it on a user-provided dataset. Several pre-trained models are ready for use in Detectron2, and for this study the Base

(Faster) R-CNN with Feature Pyramid Network" (Base-RCNN-FPN) was selected. Base-RCNN-FPN is a detector proficient in multi-scale detection with high accuracy, capable of detecting a wide array of object types.

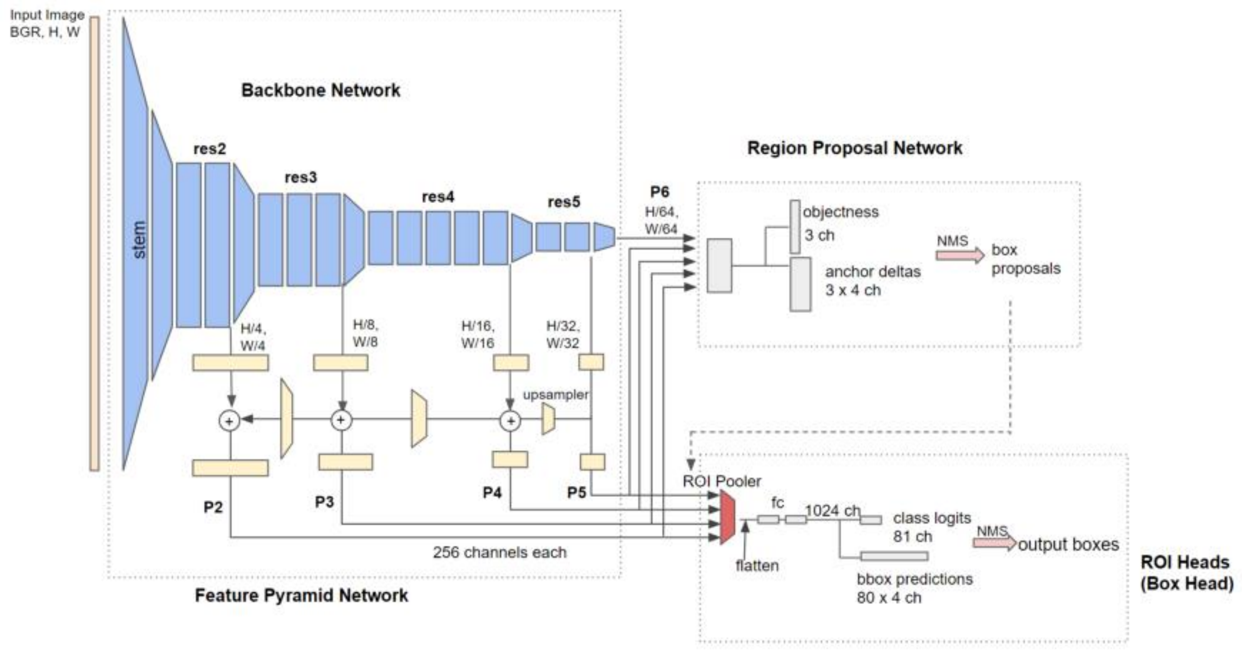


Fig 14. Base-RCNN-FPN Architecture [27]

VIII. DATA PREPROCESSING

Further preprocessing of the images had to be performed before they could be used by the models. Mainly, the images from the SS dataset were too large in their default state at 2048 x 1536 pixels. Attempting to use the original images with the CNN was not possible as the machine ran out of resources and froze during execution. Thus, scaling down the images was necessary, and all images were reduced in size to 800 x 600 pixels while maintaining the same aspect ratio of 4:3. The images were all 3 channel RGB-based. Afterwards, the two separate models had different requirements in terms of datasets.

The traditional CNN model utilizing ResNet-18 required minimal processing for the dataset. The images had to be separated into different directories depending on which classes were present (“gazelle”, “warthog”, “wildebeest”, “zebra”, “gazelle_and_zebra”, “wildebeest_and_zebra”). After this simple step, the model was ready to begin training without any further modifications.

For the R-CNN model, Detectron2 required significantly more preprocessing before the images could be used. The data needed to be labeled in such a manner where the model knew the bounding boxes of subjects for each image in the dataset. In order to perform such labeling, the dataset had to be converted to the Microsoft Common Objects in Context (COCO) format. The COCO format consists of a structured JSON file detailing the labels and metadata of all images in the dataset [30]. In order to obtain the JSON information for this dataset, the images for both the training and testing splits were hand-labeled through the *labelImg* [31] utility tool. During labeling the annotations are set for each image, such as the bounding boxes for subjects, class type, and whether the bounding box could be qualified as a difficult classification. Furthermore, key metadata information such as image dimensions is also stored in the COCO JSON.

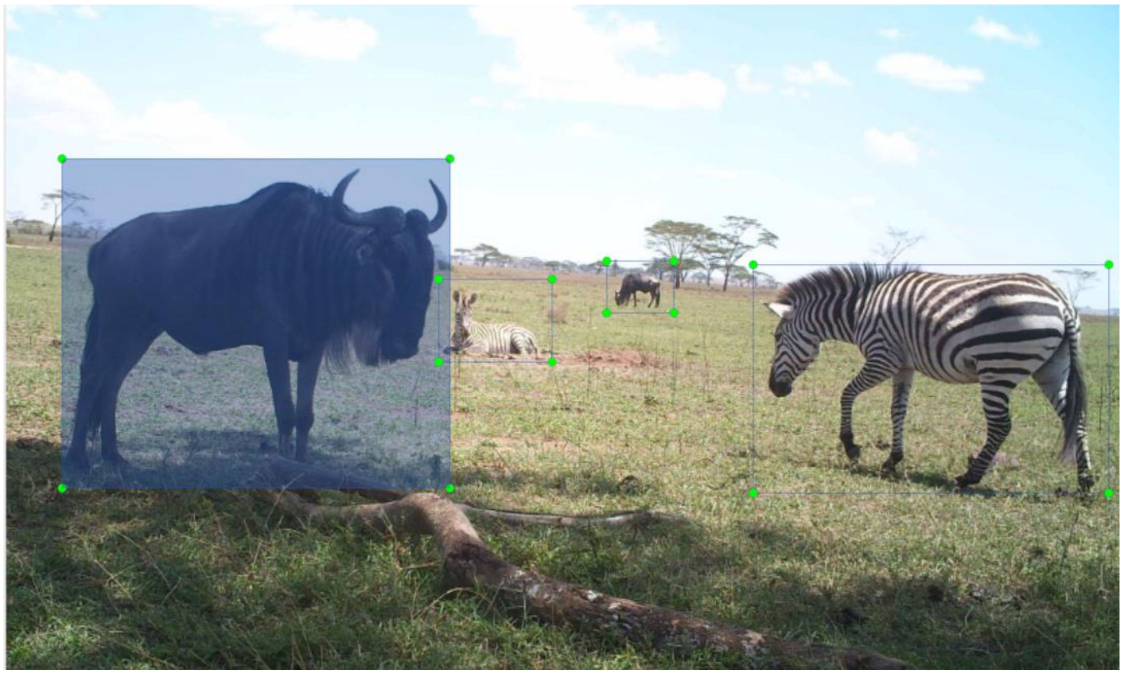


Fig 15. Labeled Image of “Wildebeest” and “Zebra” Subjects

While labeling introduces difficulty when scaling up to larger data sets, it also allows the model to extract information at a higher efficiency. The dataset contains many images that are indirect shots of subjects and have a large degree of variation between them. These variations in scenes can prove troublesome for traditional CNN models, as it can be difficult to extract the right information. The ability to provide detailed labeled images for R-CNN models allows the user to instruct the model where to focus, and can provide other information such as the distinction between subjects and if it’s a difficult classification.

IX. MODEL PERFORMANCE AND RESULTS

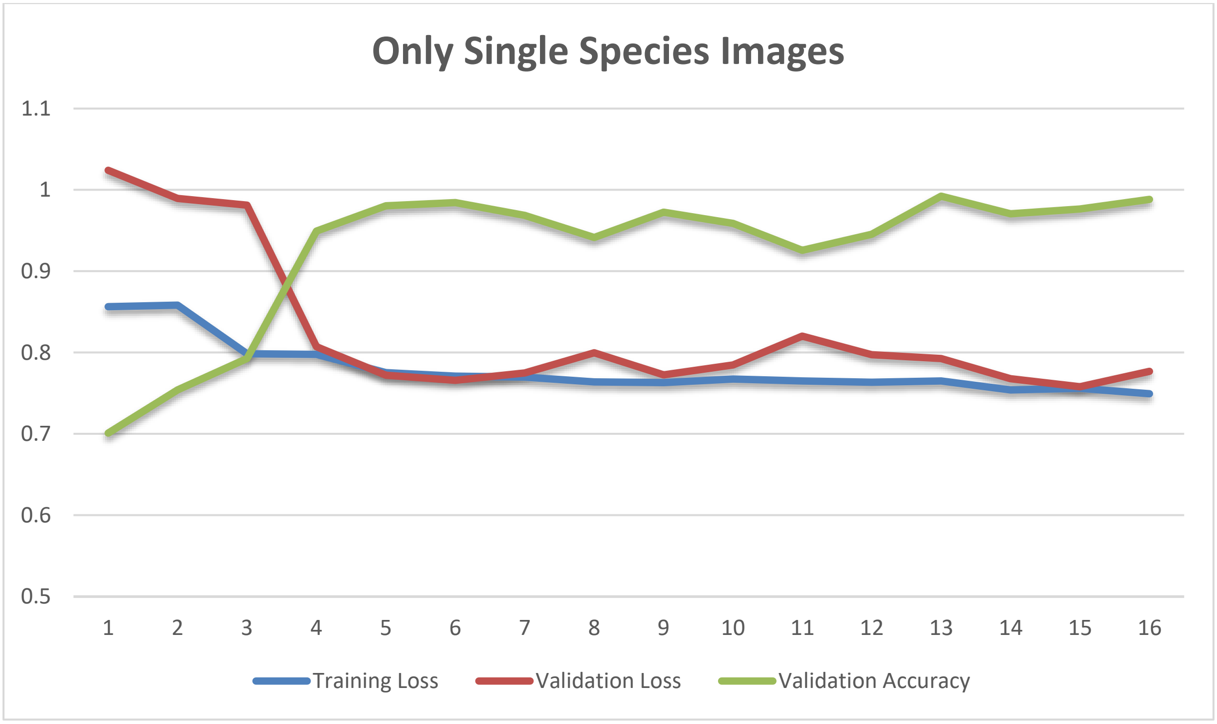
System Setup:

All the models were created using Python libraries including such as PyTorch, NumPy, and Detectron2. They were run on a machine utilizing a NVIDIA GTX 1060 GPU with 6 GB of VRAM.

A. CNN Model with ResNet-18

In order to compare the performance of the Detectron2 based R-CNN, a traditional CNN model was created to act as a base comparison. The CNN employed a pre-trained ResNet-18 model as mentioned above and was tested on two datasets, one containing individual species [Gazelle, Warthog, Wildebeest, Zebra] and the main set containing individual as well as mixed-species [Gazelle, Gazelle_and_Zebra, Warthog, Wildebeest, Wildebeest_and_Zebra, Zebra]. The reason for this was to compare the general accuracy of this model to CNNs found in previous papers. Since CNNs have proven capable at classifying photos with individual species, it was important to check if this base model was capable of the same performance. The learning rate was set at *0.001* and an optimizer function was used to update the model. The optimizer takes in the loss function to adjust the model parameters depending on the output of the loss function. Lastly, the number of epochs the model trained for was set to 16. Other epoch values greater than 16 were tested but there was no noticeable improvement. While training for more epochs can extract more information out of images, it also runs the risk of overfitting. Overfitting refers to the scenario in which the model learns patterns that commonly occur in the sample/training data but do not occur in other datasets, thus the model loses its ability for generalization as it's overfitted by the training data. All models were executed ten separate times and the results were averaged out to observe general performance.

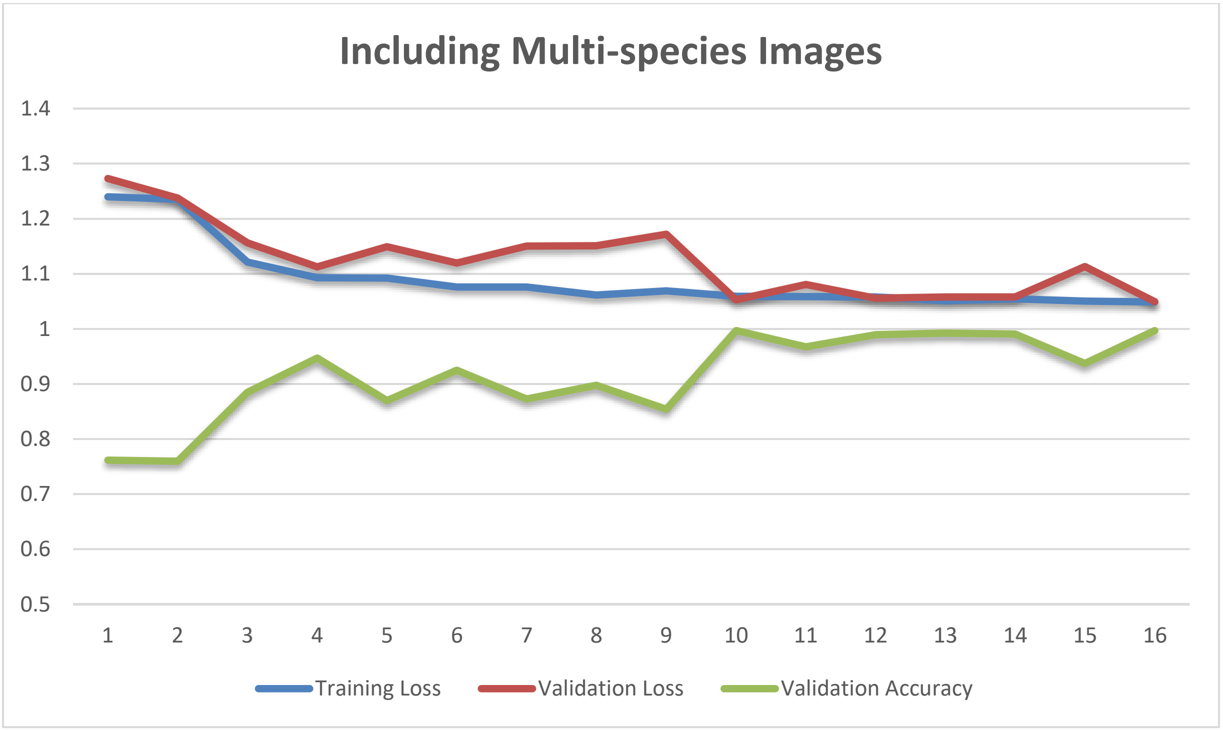
For the dataset containing individual animal species, the model performed well and returned consistent results. Average loss and accuracy improved steadily as training continued through the 16 epochs. Similar to the CNN models used in [3] and [6], this model was able to confidently distinguish between the four species as loss stayed below 1.0 and accuracy remained high at 89% for the testing dataset.



Average Loss and Accuracy on Testing Dataset	
Loss	0.820991
Accuracy	89.14%

Fig 16. Single species image results

However, results for the images containing multiple species were not as strong, as the model struggled during training and test results. The most noticeable issue is the loss, which stays above 1.00 throughout all training epochs. The loss refers to the number of errors made by the model, so the model consistently struggled during training and it shows when validating on the testing set. The loss continues to stay above 1.0 and overall accuracy plummets to just 62%.



Average Loss and Accuracy On Testing Dataset	
Loss	1.12
Accuracy	62.38%

Fig 17. Multiple species image results

It’s clear that the CNN model is capable of classifying images with a single subject, but it struggles when more than one subject is present. This is troublesome for real world studies as images often contain many animal species resulting in more class inputs and outputs that could further lower the accuracy of the model.

B. Region Based Convolutional Neural Network with Detectron2

In order to begin training the R-CNN model, Detectron2 required all datasets be in a COCO registry, so both testing and training datasets were registered. Only four classes [gazelle, wildebeest, warthogs, and zebra] were required to be registered compared to six classes in the CNN model. Since R-CNNs are based on object identification rather than image classification, images that contained multiple species did not have to be separately categorized.

The model parameters of the R-CNN model follow the same logic as the CNN model but differ slightly. Instead of epochs, there are iterations, which are the number of times a batch of data is sent through the model. In this case, it refers to both forward and backward passes. An epoch, on the other hand, refers to one instance where the network goes over the entire training set. For this experiment, the number of iterations was set to 1500. Furthermore, an additional 500 iterations were set to run on the training set. The learning rate was the same as the CNN model at .001, and batch size per image was set at 256. Lastly, a threshold score can be set to filter out low-confidence bounding boxes predicted by the model. It was set at 0.8, meaning bounding boxes scoring below 80% accuracy were discarded. The model was run 10 times and the average results of those runs are listed in Fig 18 and 19.

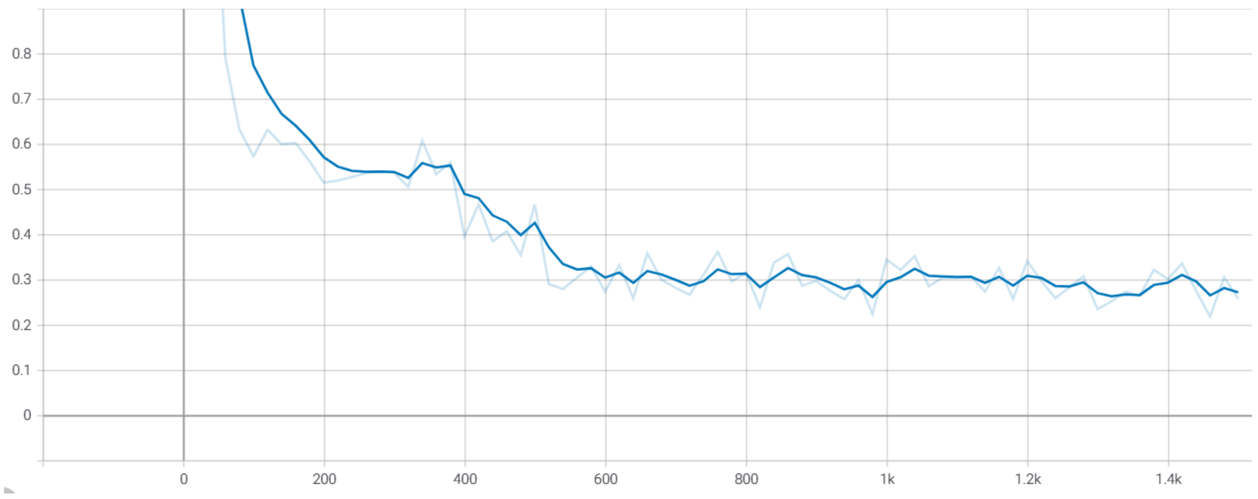


Fig 18. Average total loss of R-CNN Model

	Intersection over Union (IoU)	**Area**	**Maximum Detections (MaxDets)**	**Value**
Average Precision	0.50:0.95	All	100	0.495
Average Precision	0.5	All	100	0.844
Average Precision	0.75	All	100	0.550
Average Precision	0.50:0.95	Small	100	0.089
Average Precision	0.50:0.95	Medium	100	0.371
Average Precision	0.50:0.95	Large	100	0.621
Average Recall	0.50:0.95	All	1	0.394
Average Recall	0.50:0.95	All	10	0.549
Average Recall	0.50:0.95	All	100	0.549
Average Recall	0.50:0.95	Small	100	0.146
Average Recall	0.50:0.95	Medium	100	0.432
Average Recall	0.50:0.95	large	100	0.674

Fig 19. Results from Detectron2 Model

Evaluating the trained model on the test dataset yielded positive results. One immediately noticeable improvement is seen in the total loss. The loss of the CNN model stayed very high above 1.1, demonstrating the model made many mistakes during classification. But the R-CNN had its loss stabilize around 0.3, indicating this model to be significantly less error-prone.

Intersection over Union (IoU) is an evaluation metric that measures the difference between the hand-labeled bounding box of the test set with the predicted bounding box from the model. In object detection, it's unlikely that the coordinates of the predicted box will identically match up with the hand-labeled box, so a lenient IoU value can provide some leeway during detection.

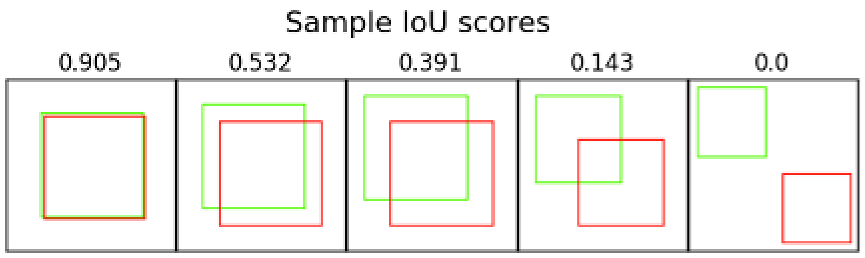


Fig 20. Sample IoU scores

The IoU value of "0.50:0.95" refers to the average precision value measured across the different IoU thresholds, incrementing by 0.05 each step i.e. (0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95).

The accuracy of the model is determined through the "Average Precision" evaluation metric from the Fig 19. results. This group of precision values states the accuracies for the different bounding box types encountered in the images. For an IoU of 0.50:0.95, across all bounding box areas, the model achieved roughly 50% accuracy. However, for this study, many images contain animals that are either camouflaged, far back in the scene, or are shot at awkward angles, so an IoU of 0.5 still provides a confident classification in these conditions. When the IoU is set to 0.5, the accuracy increases to 84.4%. Compared to the CNN model, at 62% accuracy, the R-CNN performs far better. Performance results for specific bounding box areas are also obtained for the IoU range of 0.50:0.95; small bounding box accuracy is at 8.9%, medium box accuracy is at 37.1%, and large bounding boxes have the highest accuracy at 62.1%. This discrepancy across the box sizes is expected since subjects that are closer to the camera lens tend to be clearer and more in focus compared to subjects located in the background. Even humans can struggle to properly identify objects under such conditions.

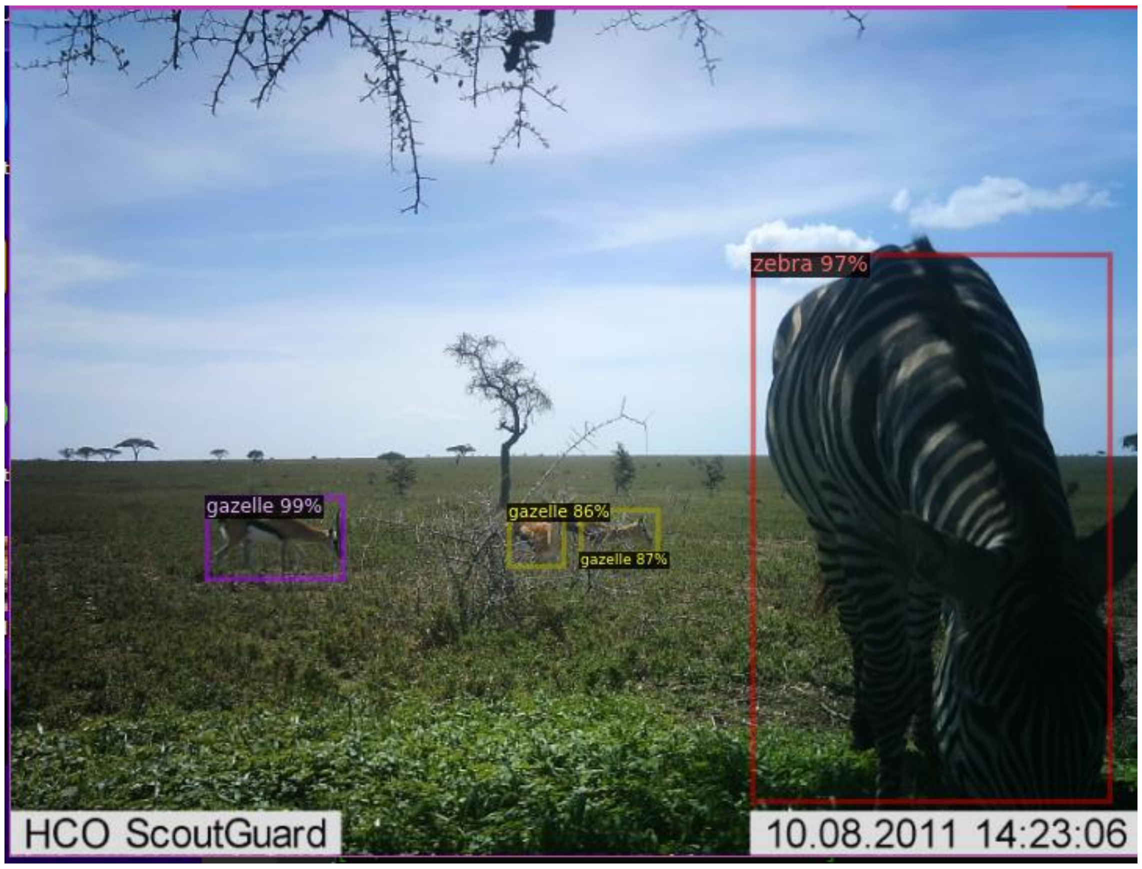


Fig 21. Model output from testing dataset

C. Detectron2: Scaling Up with Additional Elephant Class

One of the main strengths of Detectron2 is its ability to obtain strong precision results even as more class types are added to the model. Camera trap studies often vary in scale so this attribute proves useful for both small- and large-scale studies. To test this, an addition of a fifth class, Elephants, was made to observe if any significant performance drop occurred. Like the other classes, 500 images were provided for training while 60 images were set for testing validation. Both the training and testing image sets were hand-labeled with bounding boxes to detail the objects. Fortunately, the previously labeled images for the other classes do not need to be modified and can be used again as-is. The model was run 10 times and the average result of those runs is listed in Fig 23.

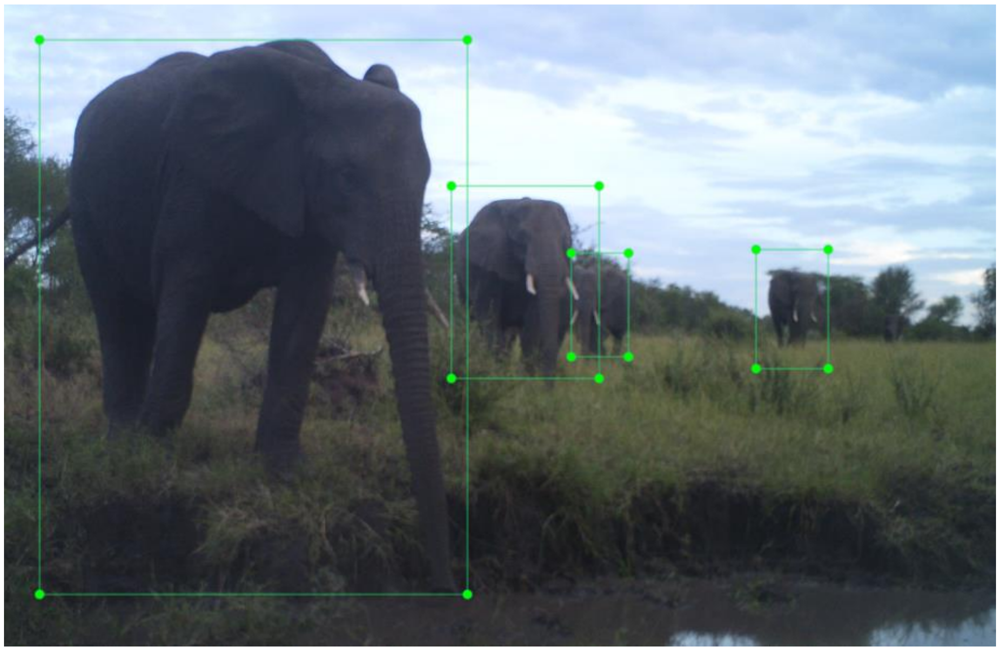


Fig 22. Labeled Image of Elephant Subjects

	Intersection over Union (IoU)	**Area**	**Maximum Detections (MaxDets)**	**Value**
Average Precision	0.50:0.95	All	100	0.516
Average Precision	0.5	All	100	0.832
Average Precision	0.75	All	100	0.590
Average Precision	0.50:0.95	Small	100	0.067
Average Precision	0.50:0.95	Medium	100	0.287
Average Precision	0.50:0.95	Large	100	0.633
Average Recall	0.50:0.95	All	1	0.406
Average Recall	0.50:0.95	All	10	0.561
Average Recall	0.50:0.95	All	100	0.561
Average Recall	0.50:0.95	Small	100	0.087
Average Recall	0.50:0.95	Medium	100	0.325
Average Recall	0.50:0.95	large	100	0.683

Fig 23. Results from Detectron2 Model with Elephant Class

The model's results with five classes are similar to what was achieved with four classes. At an IoU of 0.5, accuracy stays high at 83.2%, just 1.2% less accurate than what was achieved with four classes (84.4%). Surprisingly, as the IoU values become more stringent, the average precision scores of the model show slightly better performance with five classes. This can be seen across the IoU range of 0.50:0.95, where the model shows a 2.1% increase in accuracy. When observing results across the different bounding box areas, there is a decrease in performance for small (6.7% vs 8.9%) and medium boxes (28.7% vs 37.1%), but an increase for large boxes (63.3% vs 62.1%). This can be explained by the addition of the latest elephant class. The elephant class provides bounding boxes for an animal species with distinct size and a large shape, which can explain the performance improvement for large bounding box types. However, this advantage disappears when the object is further back in the shot. In those circumstances, there is a higher chance of the elephant appearing as another animal species or part of the background.

X. CONCLUSION

Camera traps have proven themselves to be an essential tool for performing ecological surveys, but simultaneously required significant effort in terms of manually labeling the resultant images. Image processing and machine learning techniques act as potential solutions for this issue and have been successfully used across numerous fields with impressive results. But camera trap images come with their own set of unique difficulties. Many images have a great deal of variation between one another and can be difficult to extract information from. Previous studies [3] [6] have shown that CNN models work well in identifying images with a single species but struggle when multiple species are present. Fortunately, more specialized techniques such as object detection aim to combat the difficulties encountered by traditional image classification models. The Detectron2 based R-CNN model that was tested in this paper performed well in these troublesome scenarios. When considering a dataset that contained images of both single and mixed-species images, the R-CNN model obtained 84% accuracy when compared to the CNN model with just 62% accuracy. The ability for Detectron2 to take user-provided labels for images proved to be the crucial advantage as it provides specific context on what image features the model should look for.

XI. FUTURE WORK

The Detectron2 based R-CNN model performed well, but was limited to four animal species. A greater variety of species are included in the Snapshot Serengeti project and additional progress can be made in a straightforward manner by adding labeled images of other animal species to the dataset. While hand-labeled images provide significant information to the model, they come at the cost of heavy data preprocessing. Requiring proper labeling for both the training and testing datasets is a tedious task that only gets more difficult as the dataset increases. If this study were scaled up to cover all species images from the park instead of four, there would be tens of thousands of labels that would need to be created. Hence, the reason why only four species were tested in this study is to act as proof of concept. However, from the strong results demonstrated, it would be a worthwhile investment to attempt such a survey. Since it’s possible to store the trained model, it’s feasible to train the model over an extended time period rather than all at once. This would ease the burden on the researchers providing labeled data and on the machines that train the model.

REFERENCES

[1] E. H. Fegraus, K. Lin, J. A. Ahumada, C. Baru, S. Chandra, C. Youn "Data acquisition and management software for camera trap data: A case study from the TEAM Network" *Ecological Informatics,* https://doi.org/10.1016/j.ecoinf.2011.06.003, 2011

[2] K. R. R. Swinnen, J. Reijniers, M. Breno, H. Leirs "A Novel Method to Reduce Time Investment When Processing Videos from Camera Trap Studies" *PLoS ONE* 9(6): e98881. https://doi.org/10.1371/journal.pone.0098881, 2014

[3] M. S. Norouzzadeh, A. Nguyen, M. Kosmala, A. Swanson, M. S. Palmer "Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning" *Proc. Natl. Acad. Sci.* https://doi.org/10.1073/pnas.1719367115*,* 2019.

[4] G. Harris, R. Thompson, J. L. Childs, J. G. Sanderson "Automatic Storage and Analysis of Camera Trap Data*" Ecological Society of America* Volume 91, Issue 3, https://doi.org/10.1890/0012-9623-91.3.352, 2010

[5] K. Figueroa, A. Camarena-Ibarrola, J. García, H. T. Villela "Fast automatic detection of wildlife in images from trap camera" *CIAR,* https://doi.org/10.1007/978-3-319-12568-8_114, 2014

[6] Z. Miao, K. M. Gaynor, J. Wang *"*Insights and approaches using deep learning to Classify wildlife". *Sci Rep* 9**,** 8137, https://doi.org/10.1038/s41598-019-44565-w, 2019.

[7] Y. LeCun, Y. Bengio, & G. E. Hinton "Deep learning". *Nature* 521, 436-444, https://doi.org/10.1038/nature14539, 2015.

[8] M. A. Tabak, M. S. Norouzzadeh, D. W. Wolfson, *et al.* "Machine learning to classify animal species in camera trap images: applications in ecology". *bioRxiv*, https://doi.org/10.1101/346809, 2018.

[9] F. Schindler, V. Steinhage "Identification of animals and recognition of their actions in wildlife videos using deep learning techniques" *Ecological Informatics Vol 61,* https://doi.org/10.1016/j.ecoinf.2021.101215., 2021

[10] S. Schneider, G. W. Taylor, S. C. Kremer “Deep Learning Object Detection Methods for Ecological Camera Trap Data” *Methods in Ecology and Evolution* DOI: 10.1111/2041-210X.13133, 2019

[11] J. Niedballa, R. Sollmann, A. Courtiol, A. Wilting “camtrapR: an R package for efficient camera trap data management” *Methods in Ecology and Evolution* https://doi.org/10.1111/2041-210X.12600, 2016

[12] R. Steenweg, M. Hebblewhite, R. Kays, J. Ahumada, J. T. Fisher, Co. Burton, S. E. Townsend, *et al.* “Scaling-up camera traps: monitoring the planet's biodiversity with networks of remote sensors”. *Frontiers in Ecology and the Environment* http://www.jstor.org/stable/44133645, 2017.

[13] G. Falzon, C. Lawson, K. Cheung, K. Vernes, G. A. Ballard and P. J. Fleming “ClassifyMe: A Field-Scouting Software for the Identification of Wildlife in Camera Trap Images” *Animals (Basel)* Vol. 10, Issue 1 http://dx.doi.org.libaccess.sjlibrary.org/10.3390/ani10010058, 2020

[14] A. G. Villa, A. Salazar, F. Vargas “Towards automatic wild animal monitoring: Identification of animal species in camera-trap images using very deep convolutional neural networks” Ecological Informatics Vol. 41 https://doi-org.libaccess.sjlibrary.org/10.1016/j.ecoinf.2017.07.004 2017

[15] M. A. Tabak, M. S. Norouzzadeh, D. W. Wolfson, *et al.* “Improving the accessibility and transferability of machine learning algorithms for identification of animals in camera trap images: MLWIC2”. *Ecol Evol*; 10: 10374– 10383. https://doi.org/10.1002/ece3.6692, 2020

[16] S. Schneider, G. W. Taylor, S. S. Linquist, S. C. Kremer “Past, Present, and Future Approaches Using Computer Vision for Animal Re-Identification from Camera Trap Data” *Methods in Ecology and Evolution* [Online]. Available: https://arxiv.org/abs/1811.07749, 2018

[17] N. O. Mahony, S. Campbell, A. Carvalho, S. Harapanahalli, G. V. Hernandez, et al. “Deep Learning vs. Traditional Computer Vision” *Advances in Computer Vision Proceedings of the 2019 Computer Vision Conference (CVC).* [Online]. Available : https://arxiv.org/abs/1910.13796, 2019

[18] A. Swanson, M. Kosmala, C. Lintott, C. Packer “A generalized approach for producing, quantifying, and validating citizen science data from wildlife images”, Conservation Biology https://doi.org/10.1111/cobi.12695, 2016

[19] K. Simonyan, A. Zisserman “Very Deep Convolutional Networks for Large-Scale Image Recognition” [Online]. Available : https://arxiv.org/abs/1409.1556, 2015

[20] J. Wei “VGG Neural Networks: The Next Step After AlexNet” [Online]. Available : https://towardsdatascience.com/vgg-neural-networks-the-next-step-after-alexnet-3f91fa9ffe2c, 2015

[21] M. U. Hassan “VGG16 – Convolutional Network for Classification and Detection” [Online]. Available : https://neurohive.io/en/popular-networks/vgg16, 2018

[22] S. Ren, K. He, R. Girshick, J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, DOI: 10.1109/TPAMI.2016.2577031, 2017

[23] J. Redmon, S. Divvala, R. Girshick, A. Farhadi “You Only Look Once: Unified, Real-Time Object Detection” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (CVPR), 2016

[24] A. B. Swanson, M. Kosmala, C. J. Lintott “Data from: Snapshot Serengeti, high-frequency annotated camera trap images of 40 mammalian species in an African savanna” [Online] Available: https://datadryad.org/stash/dataset/doi:10.5061/dryad.5pt92

[25] “ResNet-18 Architecture” [Online]. Available : https://www.researchgate.net/figure/Original-ResNet-18-Architecture_fig1_336642248

[26] R. Balasubramanian “Region — Based Convolutional Neural Network (RCNN)” [Online] Available: https://medium.com/analytics-vidhya/region-based-convolutional-neural-network-rcnn-b68ada0db871

[27] H. Honda “Digging into Detectron 2” [Online]. Available: https://medium.com/@hirotoschwert/digging-into-detectron-2-47b2e794fabd

[28] “Intersection over Union (IoU)” [Online]. Available: http://ronny.rest/media/tutorials/localization/ZZZ_IMAGES_DIR/iou_scores.png

[29] R. Gandhi “R-CNN, Fast R-CNN, Faster R-CNN, YOLO” [Online]. Available: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e

[30] “Detectron2 GitHub Repository” [Online]. Available https://github.com/facebookresearch/detectron2

[31] E. Hofesmann “How to work with object detection datasets in COCO format” [Online]. Available : https://towardsdatascience.com/how-to-work-with-object-detection-datasets-in-coco-format-9bf4fb5848a4

[32] “LabelImg GitHub Repository” [Online]. Available https://github.com/tzutalin/labelImg