Kathmandu University

Department of Computer Science and Engineering

Dhulikhel, Kavre

A Project Report

on

"Wildlife Monitoring"

[Code No: COMP 308]

(For partial fulfillment of 3rd year/2nd Semester in Computer Engineering)

Submitted by

Amir Bhattarai(10)

Bishal Neupane(36)

Nirav Sapkota(48)

Submitted to

Dr. Rajani Chulyadyo

Department of Computer Science and Engineering

Submission Date: 15th May, 2023

Bonafide Certificate

This project work on

“Wildlife Monitoring”

is the bonafide work of

“

Amir Bhattarai (10)

Bishal Neupane (36)

Nirav Sapkota (48)

”

who carried out the project work under my supervision.

Project Supervisor

____

Dr. Gajendra Sharma

Abstract

In this project we aim to prepare a system where, when a still photograph of a wild animal is provided, the system can distinguish, if it is any of those in our database, and if positive, then the system classifies the given input. The input should be in the format of the image. There are not many systems developed yet, for wildlife classification. Some of the approaches have been made in such a field, but many of those are limited to case specific situations.

We have planned to attain the desired objectives with the use of neural network models, which will be working on a feasibly sufficient dataset. Firstly the image obtained is fed into the neural network model to extract the features in accordance with the labels provided, which will then again work on the validation data for the re-evaluation and finally could be tested to obtain the results.

The project could present a step in the field of wildlife conservation with minimized economy and balanced manpower only for the control of the animal behavior, rather than sitting behind the cameras for recognition. This project could lead to an initiative for multiclass classification for wildlife species.

Table of Contents

Acronyms/Abbreviations

ConvNet = Convolutional Neural Network

CNN = Convolutional Neural Network

CV = Computer Vision

GPU = Graphics Processing Unit

IDE = Integrated Development Environment

VGG = Visual Geometry Group

YOLO = You Only Look Once

List of Figures

Figure **Page no.**

List of Tables

Chapter 1: Introduction

1.1. Background

Wildlife conservation and the management of human-wildlife conflicts have aroused an unnegotiable conflicting scenarios that need to be handled at the present situation with undeniable time-lag and cost-effective methods of surveillance. Though the use of still and video surveillance cameras has had a rapid growth in the recent years, the enormous amount of data generated has shown a really expensive cost in relative to the work done. The purpose of surveillance may vary widely from identification of some target animals or problem behavior to estimating abundance and distribution of species of conservation importance, but they usually share a common need, which is to identify a particular group of species. And thus the need of automated mode of approach for the resolving of aroused problems.

In recent times, machine learning methods for automated recognition of animals have been increasingly used in indefinite biological research. These technologies have been improvised to capture the high resolution images in challenging environments and have consequently led to more effective management of natural resources. However, the automated detection has been limited to a particular species, i.e. the models are somewhat situation specific.

So, an initiative from our side for a multi-classification algorithm, a detection approach for the identification of different animals, biasing on their characteristic features.

1.2. Objectives

- Develop an automated image classification algorithm, which can identify the badgers from the classified animals, included in our dataset.

- Test, refine and calibrate the image classification algorithm to identify and classify all the possible animal species, included in our dataset.
- Test, refine and calibrate the image classification algorithm that could identify the animals from a poorly captured image with a high accuracy.

1.3. Motivation and Significance

With the increased population, the human settlements have exponentially increased, which have compelled them to move near the natural habitats of the wild lives which has increased conflicts from both the sides. As seen near the settlement areas of Chitwan, where the villages are continuously visited by the wild animals, mostly by rhinos, it has resulted in quite undesired incidents. As in the case of 23rd January 2022, in an under-construction road section in Chitwan, a rhino, which was passing by from a settlement area, fell in the drainage and ultimately lost its life there (NepalNews, 2022). There are numerous of these cases which have resulted in the decreased population of endangered wildlife species. But also when the animals got violent, they have caused deaths of civilians. Similarly, in Chitwan National Park area, a nearly middle-aged woman was attacked by an elephant on 5th August 2022 (NepalNews, 2022). Including these loss of lives from both sides, these undesirable encounters have also caused unavoidable problems.

And these situations can be handled if a monitoring system is deployed in these conflicting zones. But there may be many of such areas and having people behind all these sections could be a huge economic challenge in such situations. So, we would like to present an approach to solve all these situations with a computerized mechanism where an alert is to be generated after an undesired wildlife is encountered in settlement areas. And we are proposing an algorithm for animal classification and detection which could aid in such situations.

Chapter 2: Related Works/ Existing Works

Though many recognizable works have not been done in this field, many of the case specific projects have been carried out, whose implementations have helped to solve the particular situations. But yet, some far-fetched achievement has to be secured.

2.1. Elephant Detection and Monitoring

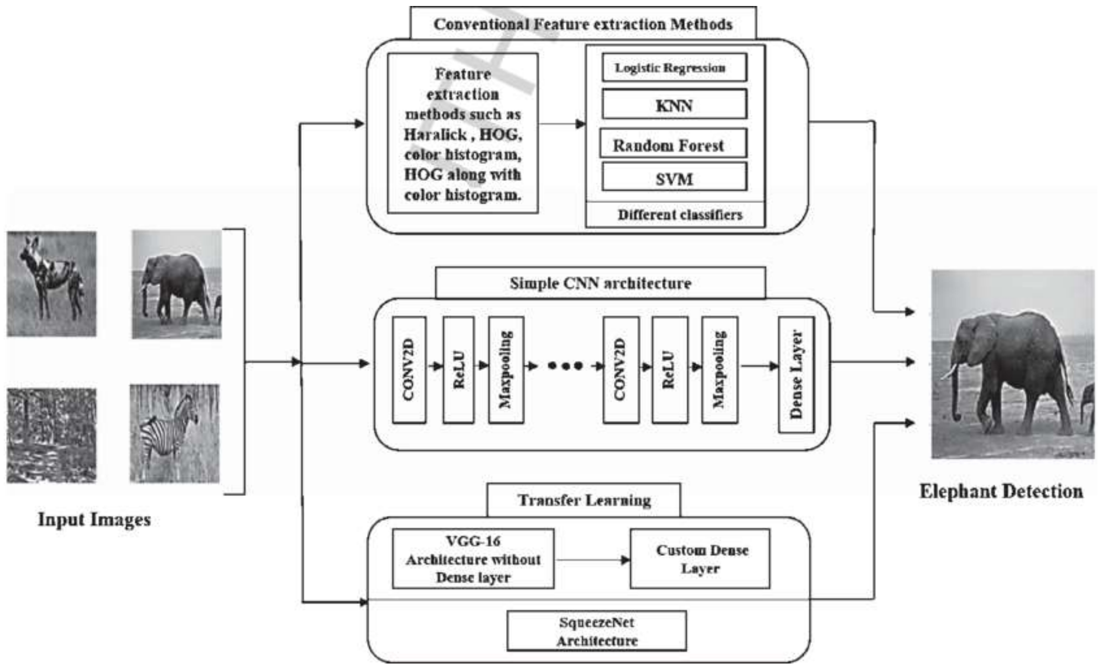


Figure 2.1. Elephant Detection Mechanism

Automated detection and tracking of elephants using color to separate the animals from the background have been successful (Zeppelzauer, 2013) and while the approach could be adapted to other species, it would not be applicable where color is absent, such as in the case of nocturnal species.

2.2. Recognition by automatic video processing

In addition to automatic recognition from still photographs, automatic video processing also has been implemented in an approach by a dairy house, (Martinez-Ortiz, Everson, & Mottram, 2013), to locate and track dairy cows. Although one of the challenges here is to be able to distinguish specific individual animals, while rejecting images that contain people and other animals.

2.3. Management of Bovine Tuberculosis

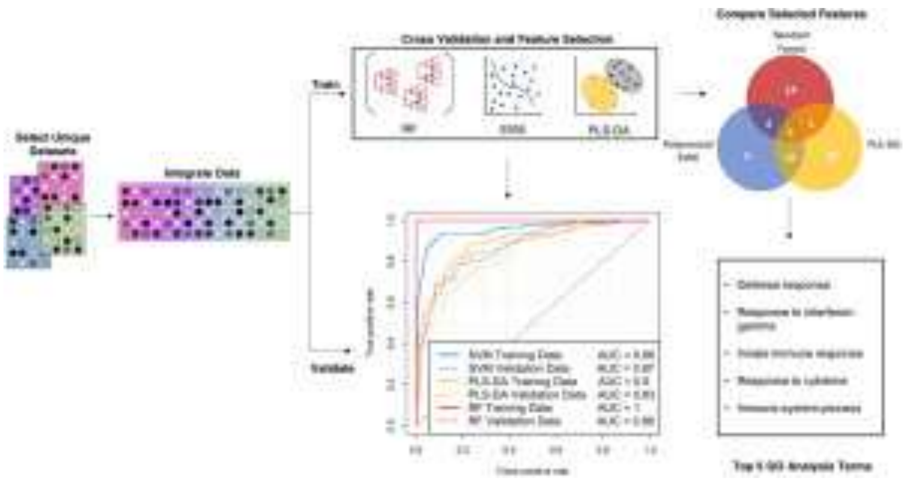


Figure 2.2. Bovine Tuberculosis Characterization

Still cameras and CCTV have been used for many years to monitor wildlife visits to farms in the UK as a part of bovine tuberculosis (Payne, Chappa, Hars, Dufour, & Gilot-Formont, 2015). European badgers (Meles meles) are identified as a potential source of infection and their presence on the field act as a direct or indirect source for the transmission. Hence, a priority is maintained for the monitoring of badger behavior in the farm environments.

Chapter 3: Design and Implementation

We have properly researched the needs and requirements which includes functional, software and hardware for the creation of the model.

3.1. System Architecture

The very basis for the implementation of our every architecture is CNN. A convolutional Neural Network is a class of artificial neural networks most commonly used to analyze visual imagery. The basis for CNN is the use of mathematical operation, ‘convolution’, in the place of ordinary matrix multiplication specifically to process pixel data, and is used in image recognition and processing. CNN are regularized versions of multilayer perceptrons. A CNN consists of an input layer, hidden layers and an output layer. Here, the hidden layers is a compilation of multiple layers that perform convolutions, typically Frobenius inner product with the common use of ReLU as the activation layer. As the convolution kernel slides along the input of the matrix for the layer, the convolution operation generates a feature map, which in turn contributes to the input of the next layer. This is followed by other layers such as pooling layers, fully connected layers, and normalization layers. The main advantage of ConvNets for many such tasks is that the entire system is trained end to end, from raw pixels to ultimate categories, thereby alleviating the requirement to manually design a suitable feature extractor.

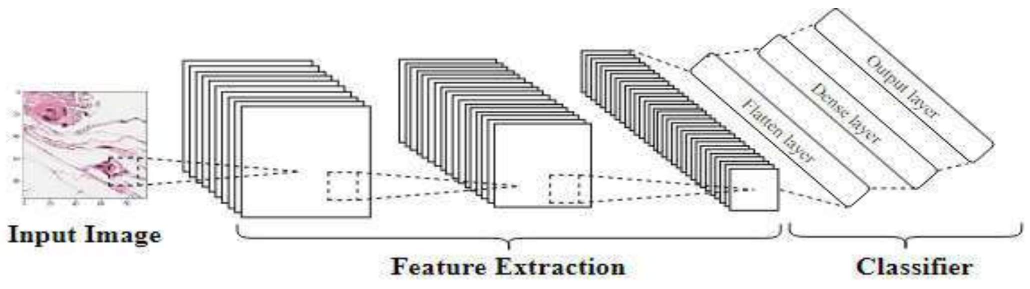


Figure 3.1. CNN Architecture

3.2. Classification

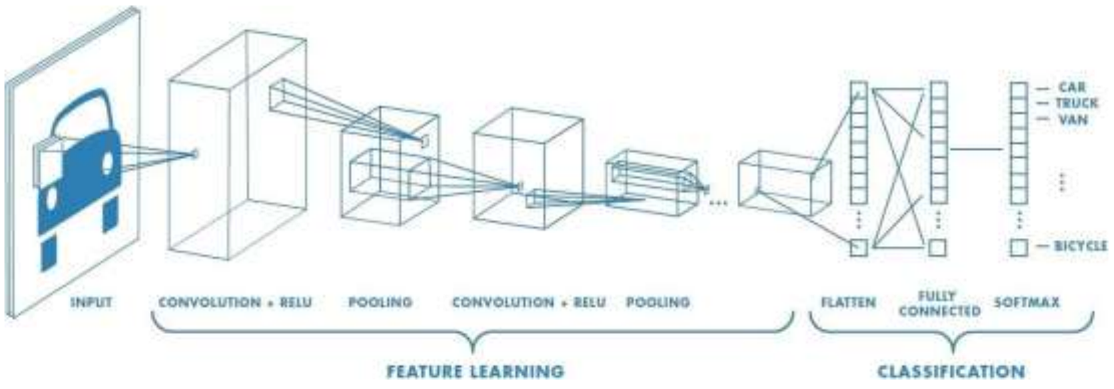


Figure 3.2. Classification with CNN Architecture

ConvNet is a Deep Learning algorithm which can take in an input image, assign weights and biases to various aspects/objects in the image and be able to differentiate one from other. The preprocessing required in ConvNet is much lower compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filter/ characteristics.

During the classification procedure, the image data is firstly splitted into train and test parts, which are then preprocessed using keras preprocessing method. And then a sequential model is created with 4 Conv2D, 2 MaxPool2D, 1 flatten and 3 Dense layers. Here, ReLU is used as an activation function for all layers except for the final Dense layer which uses softmax activation function for the classification. The model is then trained into 10 epochs with batch size of 50. And the output is shown in Figure 3.3.

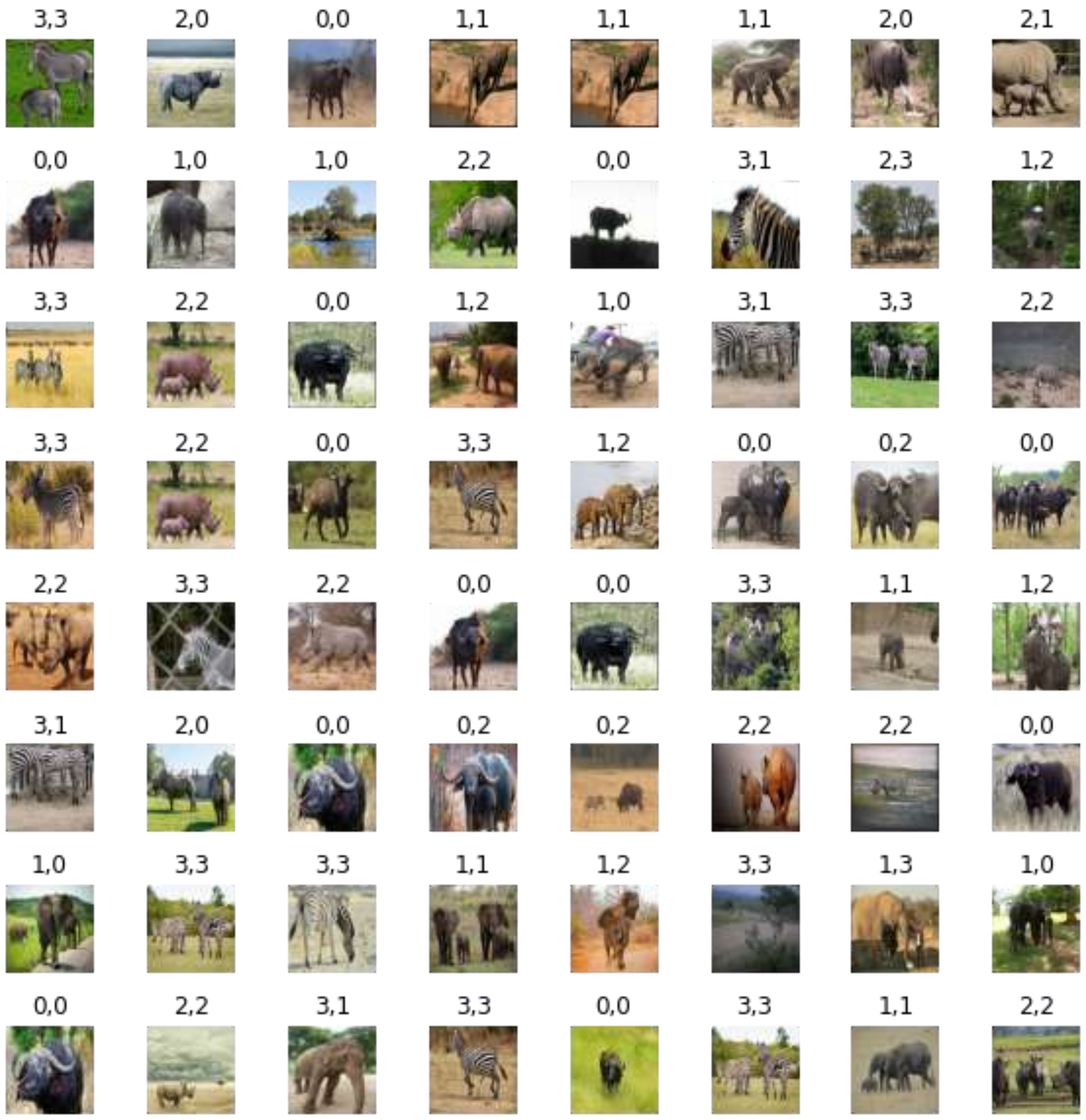


Figure 3.3. Classification on test images using Simple CNN (title showing Predicted Label, True Label)

The resulted confusion matrix is shown in Figure 3.4.

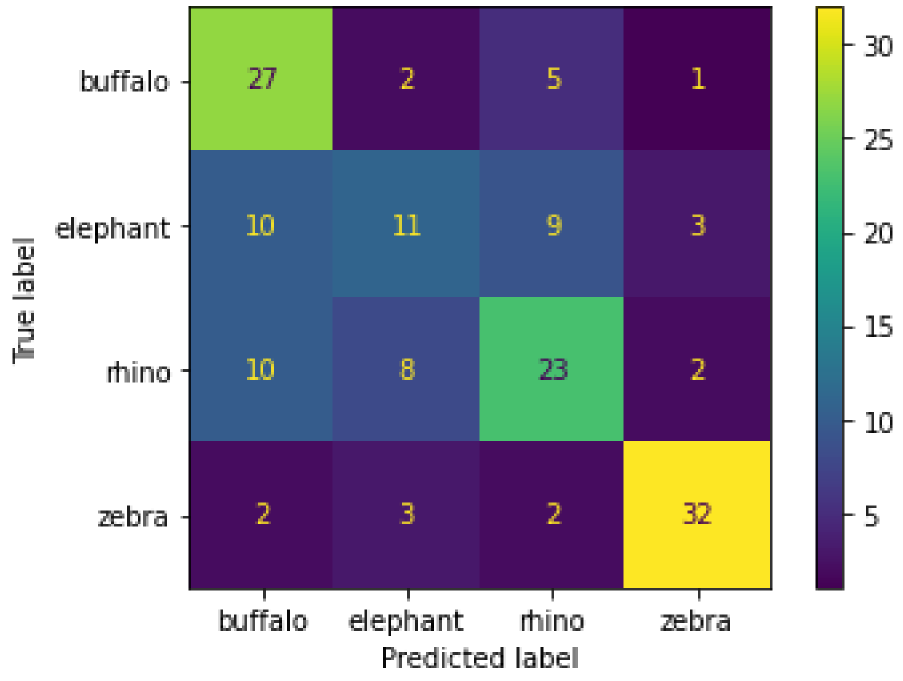


Figure 3.4. Performance Evaluation on test images using Simple CNN Model

Another method for the classification is to use the pre-trained VGG-16 model and set up according to our needs by adding some dense layers to Feature Extraction Layer. The advantage with this model is that being a bigger architecture it produced excellent result. And the output is shown in Figure 3.5.

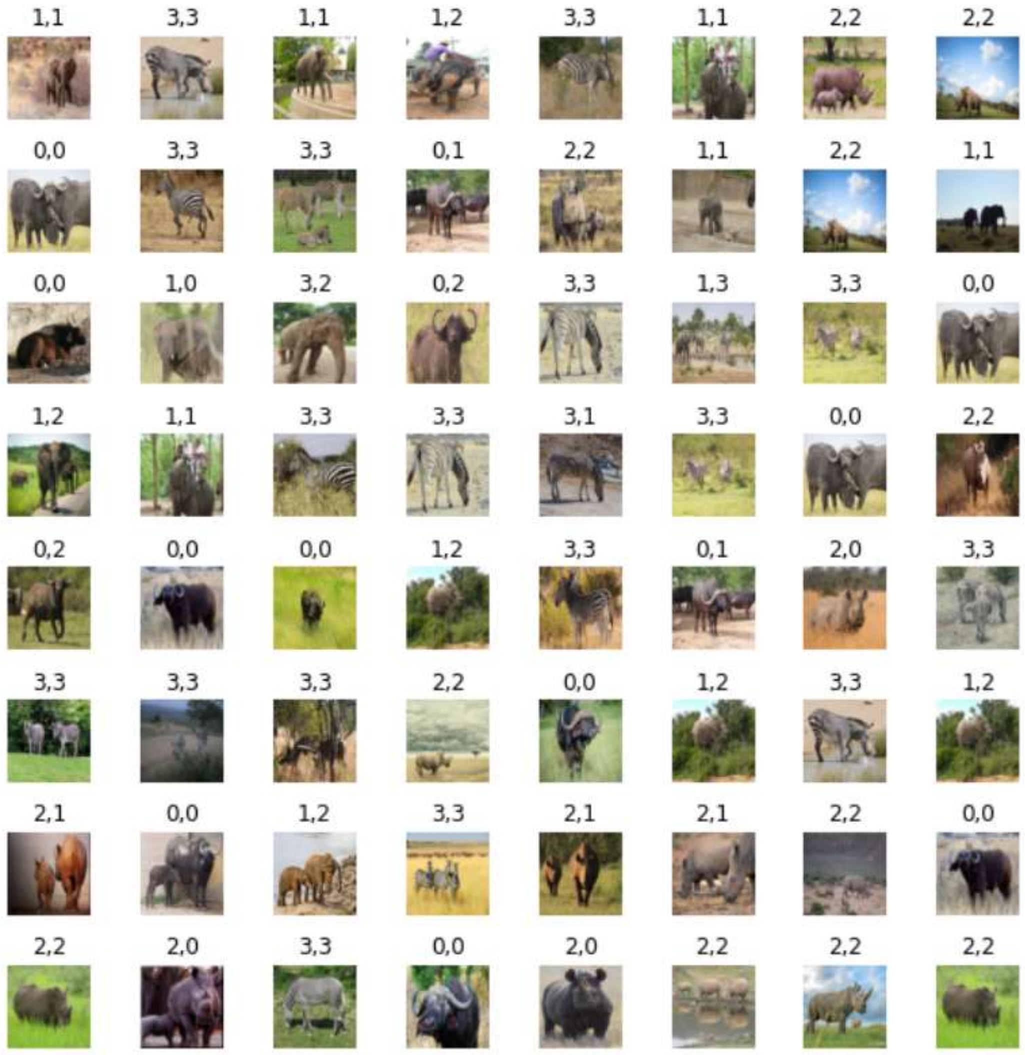


Figure 3.5. Classification on test images using VGG-16(title showing Predicted Label, True Label)

The resulted confusion matrix is shown in Figure 3.6.

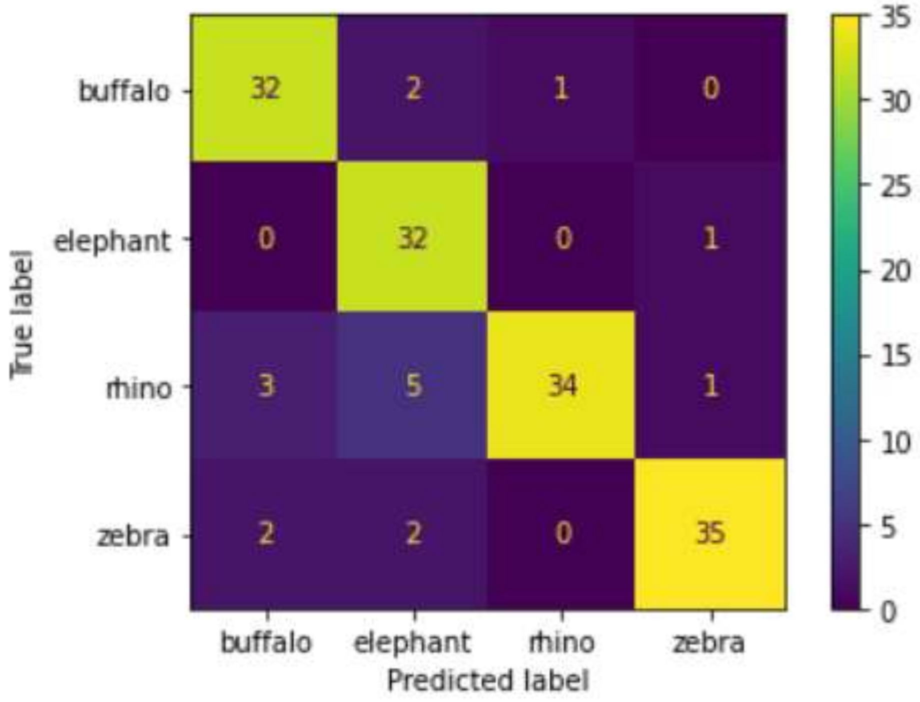


Figure 3.6. Performance Evaluation on test images using VGG-16 Model

3.3. Segmentation

Image segmentation is the process of partitioning an image into multiple segments. It is usually used for locating objects and creating boundaries. K-means clustering is one of the commonly used unsupervised learning algorithm for image segmentation. In this algorithm, pixels which have similar attributes are grouped together. The algorithm identifies different classes or clusters in the given data based on how similar it is to other data points in that same group rather than those in other groups. Here, K represents the number of clusters.

Here, we have used the preprocessed data previously obtained from during the classification procedure. And the data is passed through the method for the segmentation. The number of clusters set for during the processing is 5, which has provided a decent output.

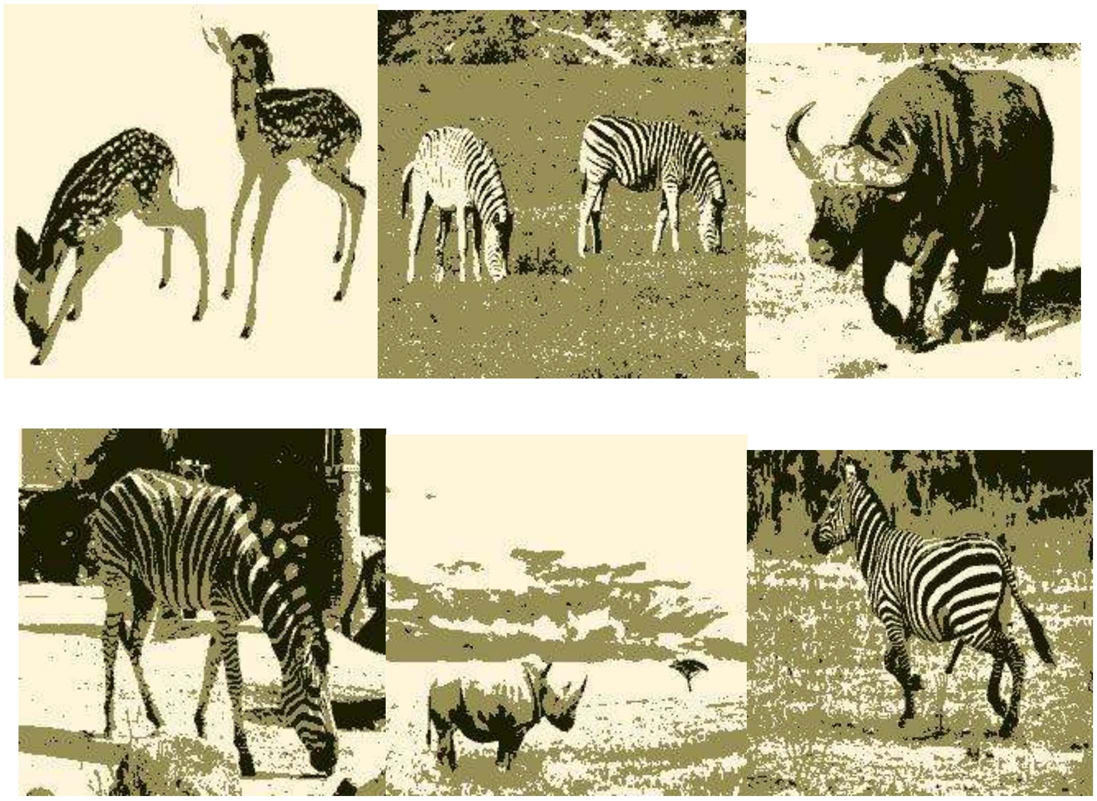


Figure 3.7. Segmentation using K-Means Clustering Algorithm

3.4. Detection

Image classification results in an output with a single class label and a probability associated with the class label prediction, which is meant to characterize the contents of the entire image, or at least the most dominant, visible contents of the image. On the other hand, object detection not only depicts the content of the image, but also the location of the object via a bounding box coordinates. Therefore and object detection algorithm allows us to input an image and obtain multiple bounding boxes and class label outputs.

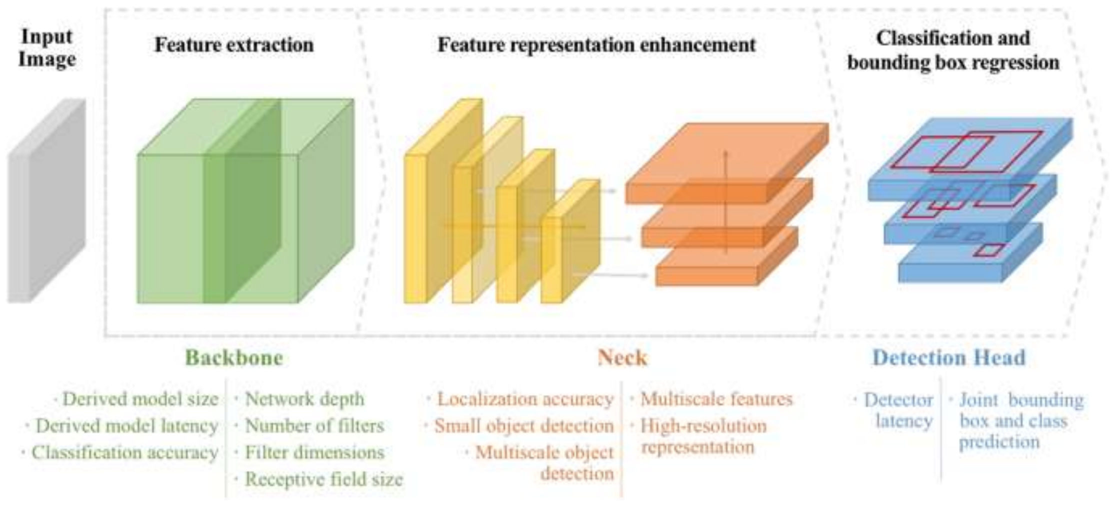


Figure 3.8. Object Detection Architecture

At the very core any object detection algorithm follows the similar pattern, regardless of the traditional computer vision or state-of-the-art deep learning. With a provided image for the object detection procedure, the output resulted is:

- a list of bounding boxes for each object in the image,
- the class label associated with each of the bounding box
- the probability/ confidence score associated with each bounding box and label

3.4.1. Multi-class object detection and Bounding Box Regression

In Multi-class object detection and bounding box regression we take VGG16 (pre-trained on ImageNet) and remove the fully-connected (FC) layer head. Then we construct a fully connected layer head with two branches. The first branch is responsible for bounding box regression whereas the second branch is responsible for making class predictions. We include this new fully connected layer to the top of the VGG16 body and fine tune the network for object detection. Its limitation is that it is able to create only one bounding box even if there are multiple objects in the images. Otherwise, the model was working very well.

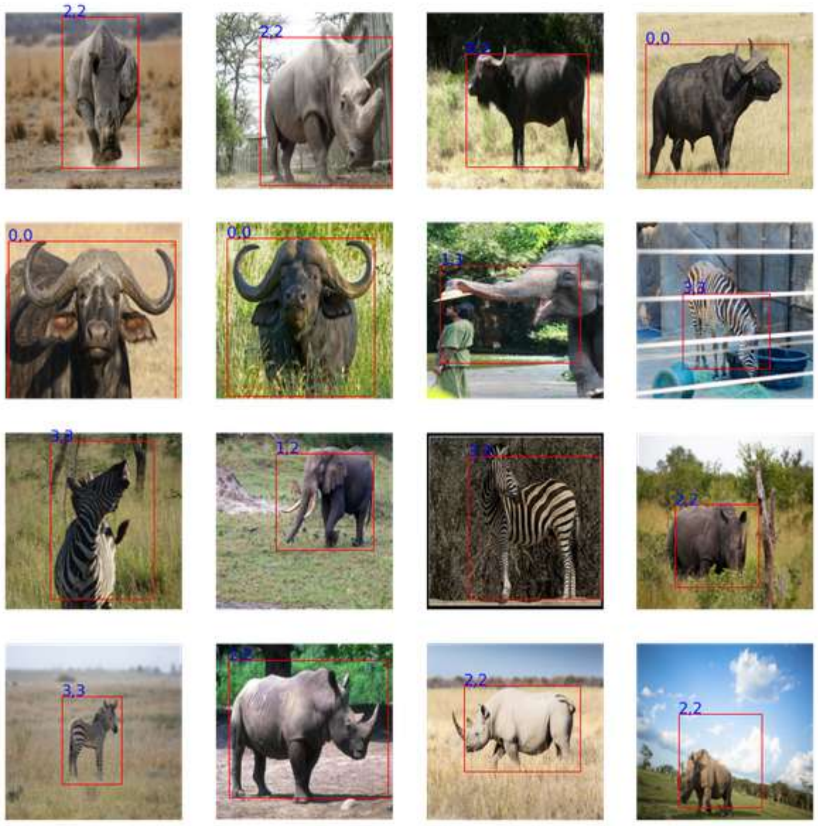


Figure 3.9. Object Detection using Regression

3.4.2. Sliding window method

The first key ingredient from is to use image pyramids (a multi-scale representation of an image). At each subsequent layer, the image is resize (subsampled) and optionally smoother. The image is progressively subsampled until some stopping criterion is met, which is normally when a minimum size has been reached and no further subsampling needs to take place.

And the second one is, as the name suggests, a sliding window is a fixed-size rectangle that slides from left- to-right and top-to-bottom within an image, the sliding window could be used to detect the face in the input image. At each step of the window we'd notice:

- Extract the ROI
- Pass it through our image classifier like linear SVM or CNN
- Obtain output predictions

Combined with image pyramids, sliding window allows us to localize objects at different locations and multiple scales of the input image

The final step is using a non-maxima suppression. When performing object detection, our object will typically produce multiple, overlapping bounding boxes surrounding an object in an image, which is normal but could pose a problem. There is only one object here and the extraneous bounding boxes need to be collapsed. The solution is to apply non-maxima suppression, which collapses weak, overlapping bounding boxes in favor of the more confident ones. The limitation of this method is that the computational cost is very high when it comes to process every ROI and give it to a classification model.

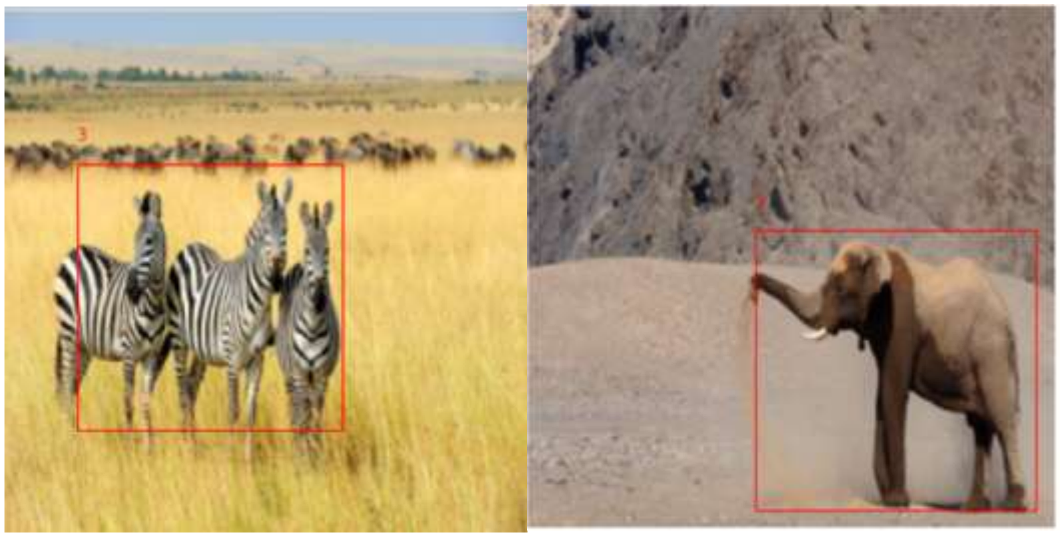


Figure 3.10. Sliding window object detection technique

3.4.3. YOLOv3

Prior work on object detection repurposes classifiers to perform detection. Instead, in YOLO, object detection is framed as a regression problem to spatially separated bounding boxes and associated class probabilities. A single neural network predicts bounding boxes and probabilities directly from full images in one evaluation. Since the whole detection pipeline is a single network, it can be optimized end to end directly on detection performance. Compared to state-of-the-art detection problems, YOLO makes more localization errors but is less likely to predict false positives in the background. Finally YOLO learns very general representation of objects. It outperforms other detection methods, including DPM and R-CNN, when generalizing from natural images to other domains like artwork.

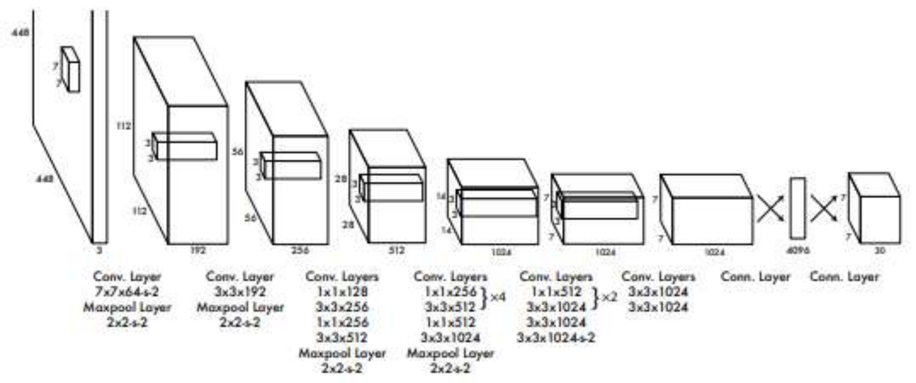


Figure 3.11. YOLOv3 Architecture

The detection network has 24 convolution networks followed by 2 fully connected layers. Instead of the inception model used by GoogLeNet, a 1x1 reduction layer followed by 3x3 convolution layers is used to reduce the feature space from preceding layers. We pretrain the convolution layers on the ImageNet classification task at half the resolution (224x224 input image) and then double the resolution for the detection. The final output of the network is a 7x7x30 tensor of predictions.

YOLO imposes strong spatial constraints on bounding box predictions since each grid cell only produces two boxes and can only have one class. This spatial constraint limits the number of nearby objects that our model can predict. The model struggles with small objects that appear in groups, such as flock of birds.

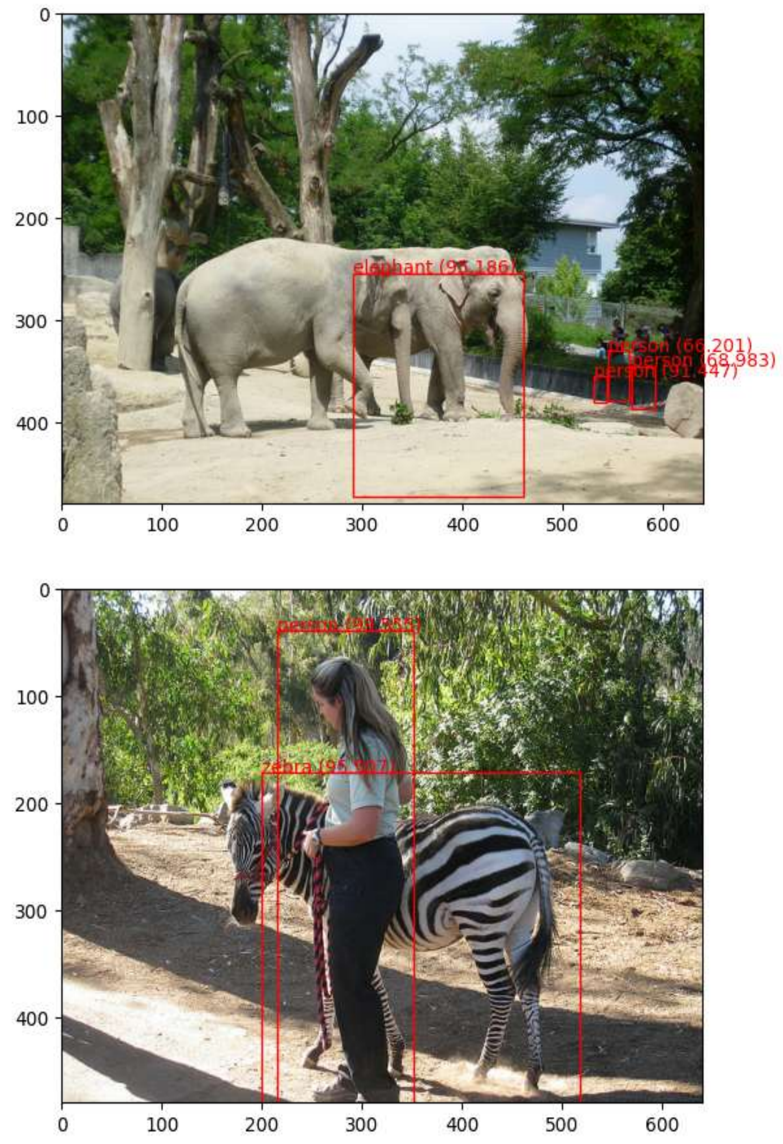


Figure 3.12. Detection using YOLOv3 Architecture

3.5. Functional Requirements

The functional requirements of the wildlife monitoring model are as follows:

1. The model must be able to identify and locate wildlife in images.
2. The model must be able to identify a variety of different species of animals.
3. The model must be able to identify animals in a versatile environment.
4. The model must be able to identify animals in a variety of different environmental situations.

Overall, the functional requirements of a CNN-based wildlife detection model should ensure that the model is accurate, efficient, and scalable for use in large-scale wildlife monitoring and conservation programs.

3.6. Use Cases

The use cases of the wildlife monitoring model are as follows:

1. It can be used to detect wildlife in images, providing a more accurate and efficient way of monitoring wildlife populations.
2. It can be used to classify wildlife in images, providing a way to differentiate various objects.
3. It can be used to segment the wildlife and environment, providing us with the objects of our interest.

So, the wildlife monitoring model can be a valuable tool for wildlife monitoring and conservation programs which contributes to the protection and conservation of the wildlife and their habitats.

3.7. System Requirement Specification

The software and hardware specifications for our project are briefed below.

3.7.1. Technology Stack

The different technology stack used in the development of the wildlife monitoring model are:

1. Programming Language: The Python programming language was used for the creation of the model. Python is one of the most popular programming languages in image detection and classification. It provides a number of libraries and frameworks that make it easy to develop image classification models.
2. Deep-Learning Frameworks: Tensorflow is an open-source software library for numerical computation using data flow graphs and Keras provides a high-level API for building neural networks, which allows users to easily create complex deep learning architectures using simple building blocks. These frameworks were used to create complex deep learning architectures i.e. Convolutional Neural Network (CNN) model. The framework also supports both CPU and GPU computing, and the wildlife monitoring model was trained.
3. Annotation Tool: Label Studio is an open-source data labeling and annotation tool used for machine learning projects. With Label Studio, users can create custom labeling interfaces for various types of data, such as text, image, and audio. The tool also supports a wide range of labeling tasks, such as classification, entity recognition, and sentiment analysis. Label Studio integrates with popular machine learning frameworks such as TensorFlow, PyTorch, and Keras, allowing us to export the labeled data in various formats that can be used directly in the models.
4. IDE : Jupyter Notebook is a web based interactive computational environment for creating and sharing documents that contain live code, equations,

visualizations, and narrative text. It was used as an Integrated Development Environment for our Python code.

3.7.2. Hardware Requirements

The hardware requirements for a CNN wildlife detection model can vary depending on factors such as the size of the model, the size of the input data, and the complexity of the tasks being performed. However, in general, deep learning models require powerful hardware to train and run efficiently.

The general hardware requirements for a CNN wildlife detection model:

1. GPU: A GPU (graphics processing unit) is essential for training deep learning models efficiently.

2. RAM: Deep learning models require a lot of memory, so a minimum of 16 GB of RAM is recommended. More RAM may be required depending on the size of the model and input data.

3. CPU: A powerful CPU with multiple cores can help speed up data pre-processing and other tasks involved in building and training a CNN wildlife detection model.

4. Storage: Large datasets and trained models can take up a lot of storage space. At least 256 GB of SSD storage is recommended.

Chapter 4: Discussion on Achievements

In this section, we discuss the different features of the project, challenges faced throughout the project and solutions to solve them.

4.1. Features

The perspective with which we chose our project was to make a comparison based study on different CNN architectures while working with image dataset. And in this study, we have come to implement various architectures for classification, segmentation and detection models. With these implementations, some of the aspects which could be listed as features are:

- Classification of wildlife which were selected for region based scenarios, regarding the context of our own locality.
- Segmentation model for the imagery dataset.
- Detection was implemented using multiple architectures, both having some drawbacks and some advantages.

4.2. Challenges and Solution

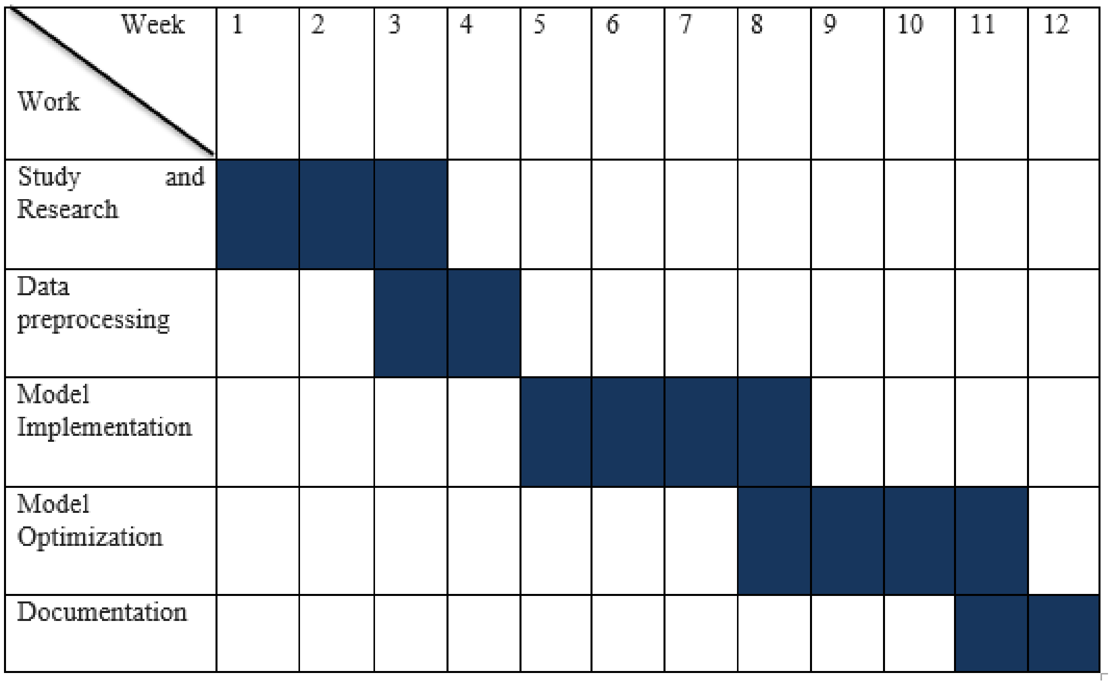
With every case there appears a scenario of challenges and we have to work through for the solutions.

- Finding the proper dataset for the relatable objective
- Implementation of various models which might be suitable for different types of data.

The solutions for these problems were obtained by thorough and continuous research, reading many articles and papers, and discussion with the teammates and supervisor.

Chapter 5: Project Planning and Scheduling

The project is scheduled to be completed in 12 weeks, allocating weeks 1 to 3 for Study and Research, weeks 3 and 4 for data preprocessing, weeks 5 to 8 for model implementation, we have allotted weeks 8 to 11 for model optimization and finally weeks 11 and 12 for documentation.



Week Work	1	2	3	4	5	6	7	8	9	10	11	12
Study and Research	■	■	■									
Data preprocessing			■	■								
Model Implementation					■	■	■	■				
Model Optimization								■	■	■	■	
Documentation											■	■

Table 5.1. Gantt chart

Chapter 6: Conclusion and Recommendation

This analysis based research project has been a base for creating a reliable application for the usage by wildlife conservation authorities, while preventing the undesired encounter between the wild lives and human. This working has been a successful study for the future workings. Though the project is solely implemented on jupyter notebook, the models implemented while using various architectures will work on the core of our future project which we believe will be a successful implementation with a UI based user application purely for the authorities, to keep an eye on the wild lives and anything that could create a chaos on their daily routines, creating unwanted scenarios for anyone. We believe to have been successful while meeting our core objective. We were able to develop a model that is able to implement classification, segmentation and multiple detection models. There might have been shortcomings, which is in the section of limited classes of objects for detection, but these will be implemented on further scenario.

6.1. Limitations

Some of the shortcomings of the model are:

- Object detection in a video or real-time detection has not be implemented
- During the training of the model for object detection, the model could only be trained for limited classes of data, due to which if an unknown object is present in the image, it could go unidentified.

6.2. Future Enhancements

If we plan to continue the development of our project in the future, some of the enhancements could be:

- Video implementation of object detection model

- Train the model in much more data so it could recognize all the possible appearances along with the noises.
- Work to place sensors and cameras to obtain real time data to be processed to keep track of undesired animals or humans on either side of the buffer area, to prevent any type of unwanted incidents.
- Work on creating a server and mobile application for the wildlife authorities for them to control the unwanted incidents.

APPENDIX

References

Martinez-Ortiz, C., Everson, R., & Motrram, T. (2013, September 11). Video tracking of dairy cows for assessing mobility scores. *European Conference on Precision Livestock Farming* (pp. 9-11). Leuven: ORE Open Research Exeter. Retrieved from http://hdl.handle.net/10871/13481

NepalNews. (2022, January 23). Rhino dies after falling in a drain. Bagmati, Nepal. Retrieved from https://nepalnews.com/s/nation/rhino-dies-after-falling-in-a-drain

NepalNews. (2022, August 5). Wild elephant killed a woman in Chitwan. Bagmati, Nepal. Retrieved from https://nepalnews.com/s/nation/wild-elephant-killed-a-woman-in-chitwan

Payne, A., Chappa, S., Hars, J., Dufour, B., & Gilot-Formont, E. (2015, October 15). Wildlife visits to farm facilities assessed by camera traps in a bovine tuberculosis-infected area in France. *European Journal of Wildlife Research*. Retrieved from https://doi.org/10.1007/s10344-015-0970-0

Zeppelzauer, M. (2013). Automated detection of elephants in wildlife video. *EURASIP Journal on Image and Video Processing*, 23.

Bibliography

Redmon, J., & Farhadi, A. (2018, April 8). YOLOv3: An Incremental Improvement. *arXiv*, 6.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016, May 9). You Only Look Once: Unified, Real-Time Object Detection. *arXiv*, 10.

Rosebrock, A. (2020, June 22). *Turning any CNN image classifier into an object detector with Keras, TensorFlow, and OpenCV*. Retrieved from pyimagesearch: https://pyimagesearch.com/2020/06/22/turning-any-cnn-image-classifier-into-an-object-detector-with-keras-tensorflow-and-opencv/

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., & LeCun, Y. (2014, February 24). OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. *arXiv*, 16.