

**EFFICIENT MODEL FOR TIGER DETECTION  
USING FASTER R-CNN**

**A THESIS SUBMITTED TO THE GRADUATE  
SCHOOL OF APPLIED SCIENCES  
OF  
NEAR EAST UNIVERSITY**

**By  
MOHAMAD ZIAD ALTOBEL**

**In Partial Fulfillment of the Requirements for  
the Degree of Master of Science**

**In  
Computer Engineering**

**NICOSIA, 2020**

**MOHAMAD ZIAD ALTOBEL**

**EFFICIENT MODEL FOR TIGER DETECTION USING  
FASTER R-CNN**

**NEU  
2020**

**EFFICIENT MODEL FOR TIGER DETECTION  
USING FASTER R-CNN**

**A THESIS SUBMITTED TO THE  
GRADUATE SCHOOL OF APPLIED SCIENCES  
OF  
NEAR EAST UNIVERSITY**

**By  
MOHAMAD ZIAD ALTOBEL**

**In Partial Fulfillment of the Requirements for  
the Degree of Master of Science  
in  
Computer Engineering**

**NICOSIA, 2020**

**Mohamad ziad Altobel: EFFICIENT MODEL FOR TIGER DETECTION USING  
FASTER R-CNN**

**Approval of Director of Graduate School of  
Applied Sciences**

**Prof. Dr. Nadire CAVUS**

**We certify this thesis is satisfactory for the award of the degree of Masters of Sciences  
in Computer Engineering**

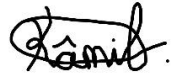
**Examining Committee in Charge:**

Prof. Dr. Rahib H. ABİYEYEV



Committee Chairman, Head of the  
Department of Computer Engineering,  
NEU

Assoc. Prof. Dr. Kamil DIMİLİLER



Committee Member, Department of  
Computer Engineering, NEU

Assoc. Prof. Dr. Melike ŞAH DİREKOĞLU

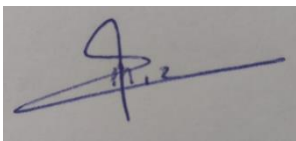


Supervisor, Committee Member,  
Department of Computer Engineering,  
NEU

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, last name: Mohamad Ziad Altobel

Signature:

A handwritten signature in blue ink, appearing to be 'M. Z. Altobel', written on a light-colored background.

Date: 1/7/2020

## **ACKNOWLEDGMENTS**

I thank the almighty Allah for enabling me to carry this study successfully. A profound gratitude goes to my supervisor for his commitment and support during this study. Lastly, I will also like to appreciate my family for their support and backup as well as every other person that has contributed to the success of this study.

**To my Family...**

## ABSTRACT

The world population of tigers has been steadily declining over the years. They have been targets for poachers over the years for sale on the black market and this has led to the extinction of 3 of the 9 major subspecies of tigers and has caused them to be declared an endangered species.

Apart from the fact that we could be deprived the opportunity to see these beautiful creatures walk the face of the earth, there are also adverse effects of this development on the ecosystem. This has made it very important that we intervene and save these creatures and consequentially, our planet.

The Tiger population does not actually need human aid to live, but it is important that they be monitored and protected. In conservation centers in India for example, Artificial Intelligence is being used to collect and analyze data on the sites. In the same line, for this project I would be using Tensor Flows Object detection API to identify Tigers in image input using a Faster R-CNN. This software can be used to analyze images captured by motion sensor cameras on conservation centers. I would also be comparing my results to results of SSD and SSD MobileNet implementations gotten from the CVWC leader board. For this project, we would be using Tensor Flow GPU instead of the regular Tensor Flow, as it would reduce the training time by a factor of 8.

**Keywords:** Artificial intelligence; Faster R-CNN; Object Detection; Tensor Flow; Tigers.

## ÖZET

Dünyadaki kaplan nüfusu yıllar içinde giderek azalmaktadır. Karaborsada yıllarca satış için kaçak avcılarının hedefi oldular ve bu, kaplanların 9 ana alt türünün 3'ünün yok olmasına ve onları nesli tükenmekte olan bir tür olarak ilan etmelerine neden oldu.

Bu güzel canlıların yeryüzünde yürüdüğünü görme fırsatından mahrum kalabilmemiz dışında, bu gelişmenin ekosistem üzerinde olumsuz etkileri de bulunmaktadır. Bu, bu yaratıklara ve sonuç olarak gezegenimize müdahale etmemizi ve kurtarmamızı çok önemli hale getirdi.

Tiger nüfusu yaşamak için aslında insan yardımına ihtiyaç duymaz, ancak bunların izlenmesi ve korunması önemlidir. Örneğin Hindistan'daki koruma merkezlerinde, sitelerdeki verileri toplamak ve analiz etmek için Yapay Zeka kullanılmaktadır. Aynı sırda, bu proje için daha hızlı bir R-CNN kullanarak görüntü girişindeki Kaplanları tanımlamak için Tensor Akış Nesnesi algılama API'sini kullanacağım. Bu yazılım, koruma merkezlerindeki hareket sensörü kameraları tarafından yakalanan görüntüleri analiz etmek için kullanılabilir. Ayrıca sonuçlarımı CVWC lider kurulundan alınan SSD ve SSD MobileNet uygulamalarının sonuçlarıyla da karşılaştırdım. Bu proje için, eğitim süresini 8 kat azaltacağından, düzenli Tensör Akışı yerine Tensör Akışı GPU'yu kullanacağız.

**Anahtar Kelimeler:** Yapay zeka; Daha hızlı R-CNN; Nesne Algılama; Tensör Akışı; Kaplanlar.



## TABLE OF CONTENTS

<b>ACKNOWLEDGMENTS</b> .....	ivV
<b>ABSTRACT</b> .....	VI
<b>ÖZET</b> .....	VII
<b>TABLE OF CONTENTS</b> .....	VIII
<b>LIST OF TABLES</b> .....	X
<b>LIST OF FIGURES</b> .....	XI
<b>LIST OF ABBREVIATIONS</b> .....	XII
<b>LIST OF EQUATIONS</b> .....	XIII
<b>CHAPTER 1: INTRODUCTION</b> .....	1
1.1 Thesis Problem .....	2
1.2 The Aim of the Thesis .....	3
1.3 The Importance of the Thesis .....	3
1.4 Limitations of the Study .....	4
1.5 Overview of the Thesis .....	5
<b>CHAPTER 2: LITERATURE REVIEW</b> .....	6
2.1 Meaning of Wildlife Conservation .....	6
2.1.1 Endangered Species .....	7
2.2 Meaning of Wildlife Conservation .....	8
2.2.1 Machine Learning Applications in Wildlife Conservation .....	10
2.2.1 Related Works .....	10
• Support Vector Machine (SVM) .....	11
• Convolutional Neural Networks (CNN) .....	13

<b>CHAPTER 3: DESIGN AND SOFTWARE TOOLS</b> .....	17
3.1 Design .....	17
3.1.1 Faster R-CNN .....	19
3.2 Software Tools .....	21
3.2.1 Tensor Flow .....	22
• High Level Object-Oriented API.....	24
3.2.2 The ATRW Dataset .....	25
• Dataset Annotation .....	26
<b>CHAPTER 4: METHODOLOGY AND DEVELOPMENT</b> .....	28
4.1 Data Acquisition .....	28
4.2 Set Up Working Environment .....	29
4.3 Data Preprocessing .....	32
4.4 Training Configuration.....	33
4.5 Training the Model.....	34
4.6 Saving and Inference.....	34
<b>CHAPTER 5: RESULTS AND DISCUSSION</b> .....	35
5.1 Test and Comparison.....	35
5.2 Constraints and Limitation of the study .....	37
<b>CHAPTER 6: CONCLUSION AND FUTURE WORK</b> .....	38
6.1 Conclusions.....	38
6.2 Future Work.....	38
<b>REFERENCES</b> .....	40
<b>APPENDICES</b> .....	42
Appendix 1: XML to CSV Converter .....	43
Appendix 2: Code to Generate TFRecord from CSV .....	45

## LIST OF TABLES

**Table 5.1:** Comparison of performances for Faster R-CNN, SSD and SSD Lite

MobileNet .....	36
-----------------	----

## LIST OF FIGURES

<b>Figure 1.1:</b>	World Tiger Population Chart .....	1
<b>Figure 2.1:</b>	Top-5 error rates of Neural Networks over 5 years compared to human Performance over the ImageNet dataset .....	9
<b>Figure 2.2:</b>	Graphical Representation of the Classification Process in SVM .....	12
<b>Figure 2.3:</b>	A typical architecture of a CNN .....	13
<b>Figure 2.4:</b>	The ReLu activation function .....	14
<b>Figure 2.5:</b>	Max pooling with a 2x2 filter .....	15
<b>Figure 2.6:</b>	The flattening process .....	15
<b>Figure 3.1:</b>	A flowchart of the system design .....	18
<b>Figure 3.2:</b>	Classifications of an image into regions with similar textures and colors by the initial CNN .....	20
<b>Figure 3.3:</b>	Faster R-CNN architecture .....	21
<b>Figure 3.4:</b>	The TensorFlow API Hierarchy .....	23
<b>Figure 3.5:</b>	Tigers within their bounding boxes .....	26
<b>Figure 3.6:</b>	A tiger with its annotated skeletal key points .....	27
<b>Figure 4.1:</b>	Splitting the dataset .....	28
<b>Figure 4.2:</b>	Downloading and extracting Faster RCNN Inception .....	29
<b>Figure 4.3:</b>	My folder structure.....	29
<b>Figure 4.4:</b>	Configuring PYTHONPATH environment variables and installing dependencies.....	30
<b>Figure 4.5:</b>	Compiling the protobuf files.....	31
<b>Figure 4.6:</b>	Compiling setup.py .....	31
<b>Figure 4.7:</b>	Command to convert the XML data to CSV .....	32
<b>Figure 4.8:</b>	Definition of the label map.....	32
<b>Figure 4.9:</b>	The label map.....	33
<b>Figure 4.10:</b>	Training the neural network model.....	34
<b>Figure 4.11:</b>	Command to generate the inference graph file.....	34

## LIST OF ABBREVIATIONS

<b>M-STrIPES:</b>	Monitoring Systems for Tigers Intensive Protection and Ecological Status
<b>PAWS:</b>	Protection Assistant for Wildlife Security
<b>IUCN:</b>	International Union for Conservation of Nature
<b>CVWC:</b>	Computer Vision fro Wildlife Conservation
<b>WWF:</b>	Worldwide Fund for Nature
<b>ATRW:</b>	Amur Tiger Re-identification in the Wild
<b>CNN:</b>	Convolutional Neural Network
<b>SVM:</b>	Support Vector Machine
<b>LBPH:</b>	Local Binary Pattern Histogram
<b>R-CNN:</b>	Region Convolutional Neural Network
<b>RPN:</b>	Region Proposal Network
<b>API:</b>	Application Programmer Interface
<b>TPU:</b>	Tensor Processing Unit
<b>IoU:</b>	Intersection of Union
<b>mAP:</b>	mean Average Precision
<b>ILSVRC</b>	ImageNet Large Scale Visual Recognition Challenge
<b>ROI</b>	Region of Interest
<b>NTCA</b>	National Tiger Conservation Authority

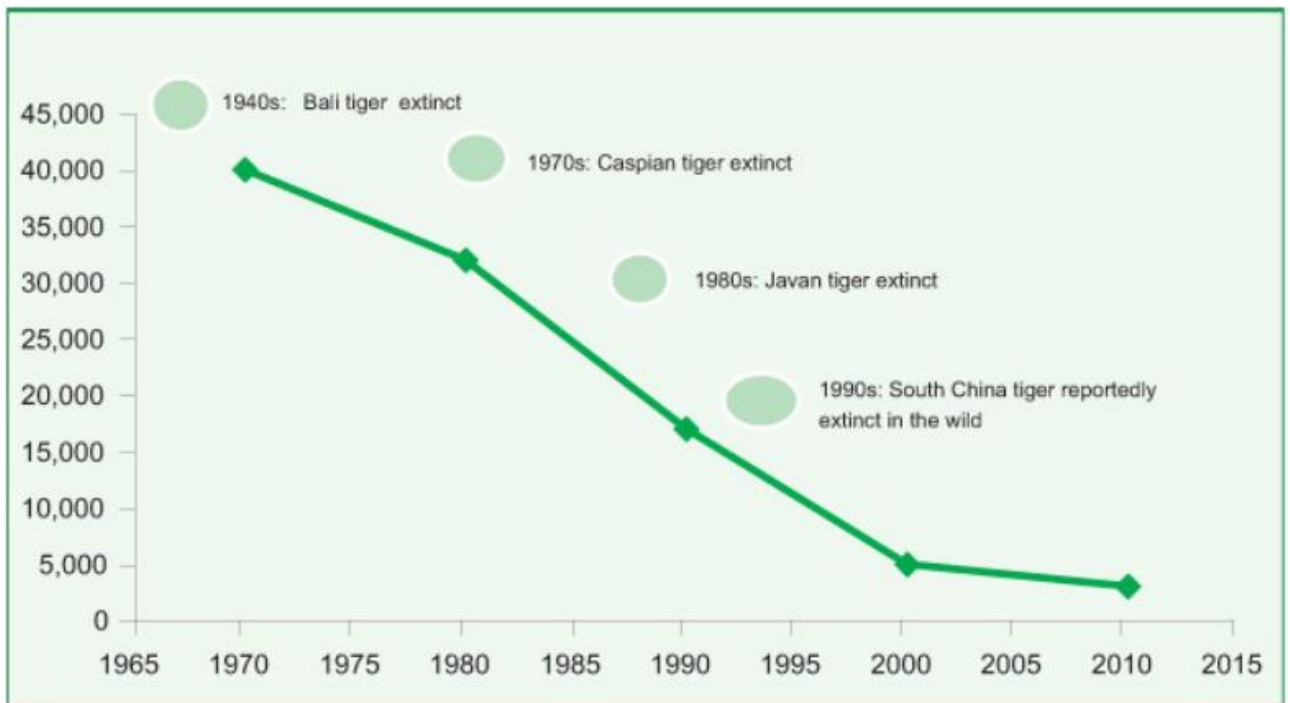
## LIST OF EQUATIONS

<b>Equation 5.1:</b>	Formula for precision .....	36
<b>Equation 5.2:</b>	Formula for IoU .....	36

# CHAPTER 1

## INTRODUCTION

Wild life conservation has become a very important topic in our society as wild life has come under threat from climate change and especially threats from human agents such as poachers. According to the UN Environment program, scientists estimate that between 150 to 200 species of plants, animals, and insects go extinct everyday as a result of climate change and human agents. This number is expected to grow if measures are not put in place to put a stop to it. One of such animals that has become endangered is the tiger. According to the World Wide Fund for Nature (WWF), there is approximately 3,900 tigers left in the world, with the Balinese Tiger going extinct in the 1940s, the Caspian Tigers in the 1970s, the Javan Tiger in the 1980s, and the South China Tiger being reportedly extinct in the wild sometime in the 1990s. Figure 1.1 shows a graphical representation of the declining tiger population between 1965 to 2010.



**Figure 1.1:** World tiger population chart

In a bid to slow the decline of the tiger population and save the species, wild life conservation centers have been working for years. However, there is still a lot of work left to be done, as even though these measures have slowed down the decline in the tiger population, the population continues to decline.

As one of the measures to help in this work, Conservation centers in India for example have started to employ Artificial Intelligence to help monitor the tiger population and all threats to their habitat, so as to help the scientists and experts gather and analyze data efficiently and develop strategies to help in saving and protecting the tiger population.

The work that is done by these wild life conservation centers is very important because these mass extinctions of species causes minor imbalances in the ecosystem that has a slow but dangerous effect on the planet. As an example, the extinction of apex predators in an ecosystem can cause an unchecked growth of the population of their prey, which would consequently put a major strain on the availability of resources such as food for their population.

## **1.1 Thesis Problem**

One of the major steps in the work to reduce the decline of the tiger population is to monitor the tiger population and all threats to them and their habitat. To be able to do this, methods that use artificial intelligence can be employed. As we well know, it would be difficult, costly and dangerous to always have people live in the wild and monitor the population.

In this thesis, I would be proposing a system that involves the uses of motion tracking cameras to take still photos of the wild. These photos can then be sent to a system that runs a convolutional neural network to classify photos that contain tigers in them and gives us these pictures to be studied and analyzed. This would totally eliminate the risk and cost of having people go into the wild and take pictures or footage to be analyzed.



Some systems have already been designed to solve the problem, but there are still improvements to be made. In this thesis, I would be designing a system with more mAP (mean average precision) and would also be using the TensorFlow-GPU instead of the regular TensorFlow, so as to reduce the training time of the convolutional neural network.

## **1.2 The Aim of the Thesis**

The aim of this thesis is to design a convolutional neural network that would be able to identify tigers in still images that would be captured from motion sensor cameras. This would go a long way in helping scientists capture as many pictures as possible from the wild, analyze the pictures, and return only pictures containing the subject (In this case, tigers) therefore helping them to properly monitor the population and any possible threats that may come to them.

This would also help to reduce the cost of having to hire extra staff to go into the wild and monitor the species while also making sure that no one has to be put in danger in the process. This thesis serves as a bridge between artificial intelligence and the noble work that is being done by scientists and wild life experts to save the tiger population and consequently, our planet.

## **1.3 The Importance of the Thesis**

Climate change is a growing issue in our world today that has caused countless debates between people that believe it is a problem and people who think it is not a real situation. However, in between all of this, it is obvious that our planet is changing. Sea levels are rising, temperatures are climbing, natural disasters are becoming more frequent, and a lot of animal species are finding it more difficult to survive. This has seriously upset the balance of the ecosystem and has in turn caused even more problems for us. Scientific analysis points to a few causes for these growing levels of extinction of species.

- Deforestation, which has caused the destruction of the natural habitats to a lot of animal species
- Poaching, which has made some of these animals considered to be “exotic animals” to be targeted because of their unique features
- Human agents, such as carbon emission, oil spills, waste disposal, etc; which has destroyed the habitats of these animals, poisoned their water sources, and caused droughts.

Much of the work that is being done by wild life experts has considerably slowed down some of these effects, however it is obvious that more work still needs to be done to solve the problem. Using artificial intelligence to automate some of the tasks involved in the work these experts do would help to solve the problem even faster and with a lower cost.

This thesis focuses on the aspect of monitoring and data analysis, helping the experts to gather as much data as possible on the field and to analyze it as fast as possible to better inform them on how to develop strategies. There are already some systems that have been put in place to help in this task. For example, the All-India Tiger estimation exercise of 2018 employed the use of an application called *M-STrIPES* (Monitoring Systems for Tigers-Intensive Protection and Ecological Status). Another artificial intelligence system called *PAWS* (Protection Assistant for Wildlife Security) was used to help in the fight against poachers in Africa back in 2016. Systems like this have gone a long way to help in the fight to save these endangered species.

#### **1.4 Limitations of the Study**

One of the limitations of the study is the fact that I do not have access to an actual wild life conservation center to test the system on the field. However, I would be running tests using already captured pictures of tigers and seeing if the system can identify the tigers within the pictures.

Also, the work requires a Linux based working environment with some python packages. The TensorFlow Object detection API also requires the use of the specific directory structure it has in its GitHub repository, so none of that can be altered.

## **1.5 Overview of the Thesis**

This thesis would be structured to first of all give a literature review in the second chapter, discussing the work that has already been done in line with artificial intelligence in wildlife conservation and what progress has been made. I would be discussing some existing systems such as *M-STrIPES*, *PAWS*, and most of the work that has come from the efforts of the CVWC (Computer Vision for Wildlife Conservation) conventions.

In the third chapter, I would then be discussing the branch of artificial intelligence I am employing here, which is neural networks. I would also discuss specifically convolutional neural networks, its steps, and why it is the method of choice for this project. I would also look at other computer vision applications in wild life conservation that use this method.

In the fourth chapter, I would discuss the software tools I would be using in the project, why I chose them, and how they are expected to affect the results I would be getting in the project.

In the fifth chapter, I would be giving a walk through of my development process and showing the steps I take in the software development process. I would also discuss any problems I have along the way.

In the sixth chapter, I would be showing and discussing my results of the process. In the seventh chapter, I would, I would be making my conclusion and discussing any improvements that can be made for future work in the field.

My code would be added in the appendix.

## CHAPTER 2

### LITERATURE REVIEW

Wildlife conservation efforts have become more popular and obvious in the 20<sup>th</sup> to 21<sup>st</sup> century, as the effects of imbalances in the ecosystem are becoming more obvious. Also, the unchecked growth of industrialization and the lack of regulation on its methods have caused a very quick negative effect on our ecosystem. With this new popularity of the topic also came the debate about what wildlife conservation truly means and entails.

In this literature review, we would first of all be looking at two main topics. The first topic would be the meaning for wildlife conservation and what it entails, as this would help us properly understand how machine learning can be introduced to the process. We would also briefly discuss what makes a species endangered and the criteria with which the “endangered” status is decided. Finally, we would be discussing early introductions of artificial intelligence and machine learning into the field of wildlife conservation. We would then focus on computer vision and the best methods so far to serve this purpose.

#### 2.1 Meaning of Wildlife Conservation

The term *wildlife* is the simple part. One overarching definition of wildlife (Usher, 1986) is that it refers to non-domesticated species of animals, plants and microbes. This implies that they are species that live, feed and propagate without any form of human intervention. Although this definition also covers plant and microbe species, most of the work that has and is being done by wildlife conservationists is focused on animal species, as they are open to much more threats than the others.

The next, and probably the most important part of the topic is *conservation*. The major task here is understanding what conservation entails, and what measures are necessary. There has been a number of perspectives discussed (Usher, 1973) ranging from a *non-interventionist* point of view (Margalef, 1968), that implies that any and all form of human

intervention is forbidden in conservation, to an *interventionist* point of view (Singh, et al, 2004), that implies that conservation is a positive attempt to set up stable and productive ecosystems. Although these points of view seem compelling in their own way, it has become clear that there isn't a fully formed philosophy of wild life conservation. The efforts and strategies seem to move back and forth with different situations.

In the work done to save endangered animal species, the *interventionist* point of view has been proven to give the best results, as the threats that these animals face have continued to grow. Poachers and unchecked deforestation are two major problems that animal wildlife conservation aims to solve. With tigers, continuous degradation of their habitat and high levels of poaching are major factors in the decline of their population. According to the Wildlife Protection Society of India (WPSI), a third of the tiger deaths of 2019 were caused by poachers. A similar amount were also lost to poachers back in 2018.

In line with this *interventionist* approach to wildlife conservation, some steps that have been taken to save the tigers include

- Close monitoring of the tiger population and all threats to them and their habitat
- Building range capacity for the tiger population to thrive
- Analyzing them and conducting research to further inform strategy decisions

In the attempt to boost the efforts of wildlife conservationists in the areas discussed above, it has become clear that artificial intelligence would serve as a powerful tool in the work being done. Artificial intelligence would greatly aid in the analysis of data and in the monitoring of the population and any threats to their habitat.

### **2.1.1 Endangered Species**

The focus of this thesis is on an endangered species - the tiger. The IUCN currently has the most comprehensive directory of endangered species in the world, and the tiger was put on this list - and declared endangered - in 1986. According to the IUCN, a species is considered to be endangered based on the following criteria:

- A 50% to 70% decline in population size in the last 10 years
- A population area of below 500km<sup>2</sup>
- An adult population size of less than 2,500
- A restricted population of 250 adults, or;
- A prediction based on statistical analysis that says the species would go extinct in 20 years.

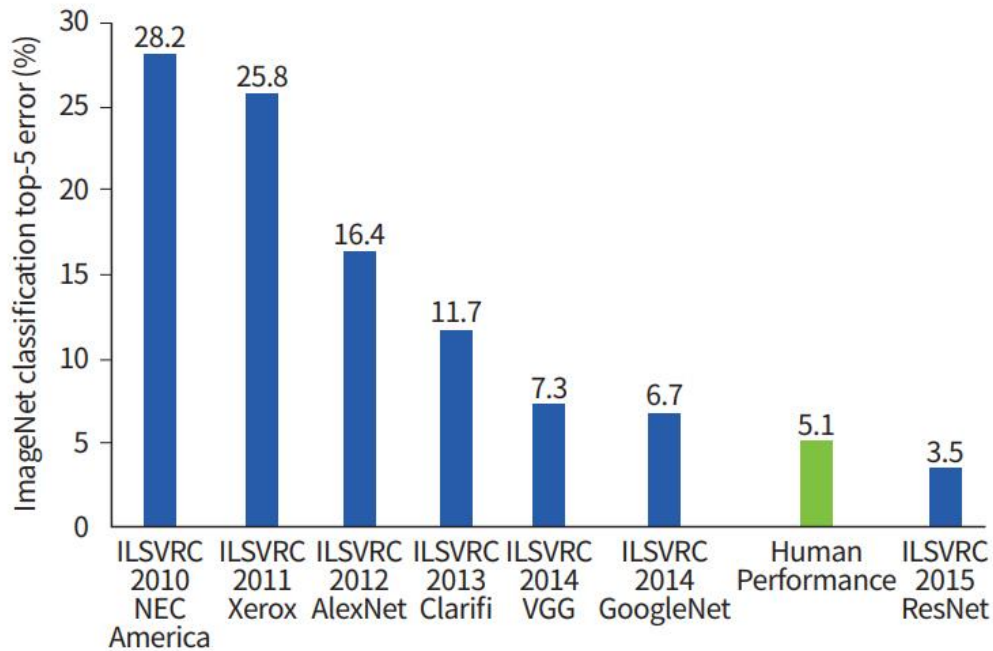
The Tiger population is now estimated to be about 3,900 total, which is only about 7% of its recorded historical population.

Some factors that have caused species to go extinct or become endangered are:

- Long reproductive cycles - long gestation periods, low numbers of offspring born, and long maturity time before offspring can mate
- Large home range - Large area needed for livelihood
- Specific habitat requirements - special conditions needed for the species to survive
- Interference of human agents - poachers, deforestation, etc

## **2.2 Machine learning and Wildlife Conservation**

Two very important aspects of wildlife conservation are data analysis and wildlife monitoring. Machine learning has proven to be one of the best tools in data analysis with its ability to recognize patterns, and powerful automation with high levels of accuracy that allow it to be used on very large amounts of data. The field of computer vision has also grown due to extensive research. A human top-5 image classification error rate on the ImageNet dataset was set to 5.1% (Russakovsky et al, 2015), while ResNet has a top-5 classification error rate of 3.57% (He et al, 2016). The top-5 rates for some major Neural Networks are shown in Figure 2.1 between the years 2010 to 2015, as well as the top-5 rate for human performance.



**Figure 2.1:** Top-5 error rates of Neural Networks over 5 years compared to human performance over the ImageNet dataset according to the ImageNet Large Scale Visual Recognition Challenge (ILSVRC).

This progress in the abilities of Machine learning has caused it to be successfully incorporated into a large and growing number of fields. This results also imply that it is safe and even recommended to apply machine learning to Wildlife Conservation.

This incorporation of machine learning into wildlife conservation is already happening in places like India for example. Computer vision is already being used to monitor animal population and habitat, and monitor poaching incidents (Thangarusa et al, 2019; Chen et al, 2014). Some more advanced systems are even able to identify, count and even specify the activities of the animals in an image (Norouzzadeh et al, 2017). Computer vision and machine learning techniques are also applied for sea turtle conservation. (Attal and Direkoglu, 2019) use CNN and Bag of Words for species classification. Whereas (Badawy and Direkoglu, 2019) utilize Faster R-CNN for sea turtle detection. In this thesis, I would be focusing on the applications of computer vision for the purpose of monitoring tiger populations.

### **2.2.1 Machine Learning Applications in Wildlife Conservation**

There are a number of AI systems already being used for wild life conservation, such as PAWS (Protection Assistant for Wildlife Security) and M-STrIPES (Monitoring Systems for Tigers Intensive Protection and Ecological Status).

PAWS is a system designed at the university of Southern California that uses machine learning to predict the behaviour of poachers and the routes they are most possible to take. This system helps to optimize the use of resources in patrols, as patrolling is still the most efficient way to prevent poaching (Fang et al, 2016). It was first tested in Uganda in 2014 and is now in regular use in Indonesia.

M-STrIPES is a system that is commissioned for use in India's tiger reserves by the NTCA (National Tiger Conservation Authority). It is a software that consists of protocols for recording patrol routes, law enforcement, recording trespassing and wild life crimes, etc.

### **2.2.2 Related Works**

There are some public methods for object detection such as: YOLO and SSD

YOLO (You Look Only One) is an effective object detection method and here I will give briefly mention about it. It is an algorithm based on regression in lieu of choosing interesting sections of the image. In running the algorithm for one time it predicate bounding boxes for each class of the object using center of the class, height, width, a value to correspond the class of the object and the probability for expecting an object in this box.

It divide the image into cells using 19x19 grid each one predicting 5 bounding boxes and this a result of 1805 bounding boxes for every image. but most of them will not contain an object and that why it give the predicting value PC which helping to remove the useless bounding boxes for the image and the bounding boxes with the high pc value will serve us to find the objects.



SSD (Single Shot MultiBox Detector for real-time processing) detection method for object detection in real-time , it consist of 2 parts ,the first one for feature map extraction and the second one Applying the convolutional neural network to detect the object ,for the feature extraction aim it use the VGG16 then use the conv 4-3 layer to detect. It use the class score and the location in a simple way using convolutional filters it use the 3x3 kernals for each section to make the predictions after extracting the feature map ,and there will be a 25 channels output to 21 scores for each class and one bounding box.

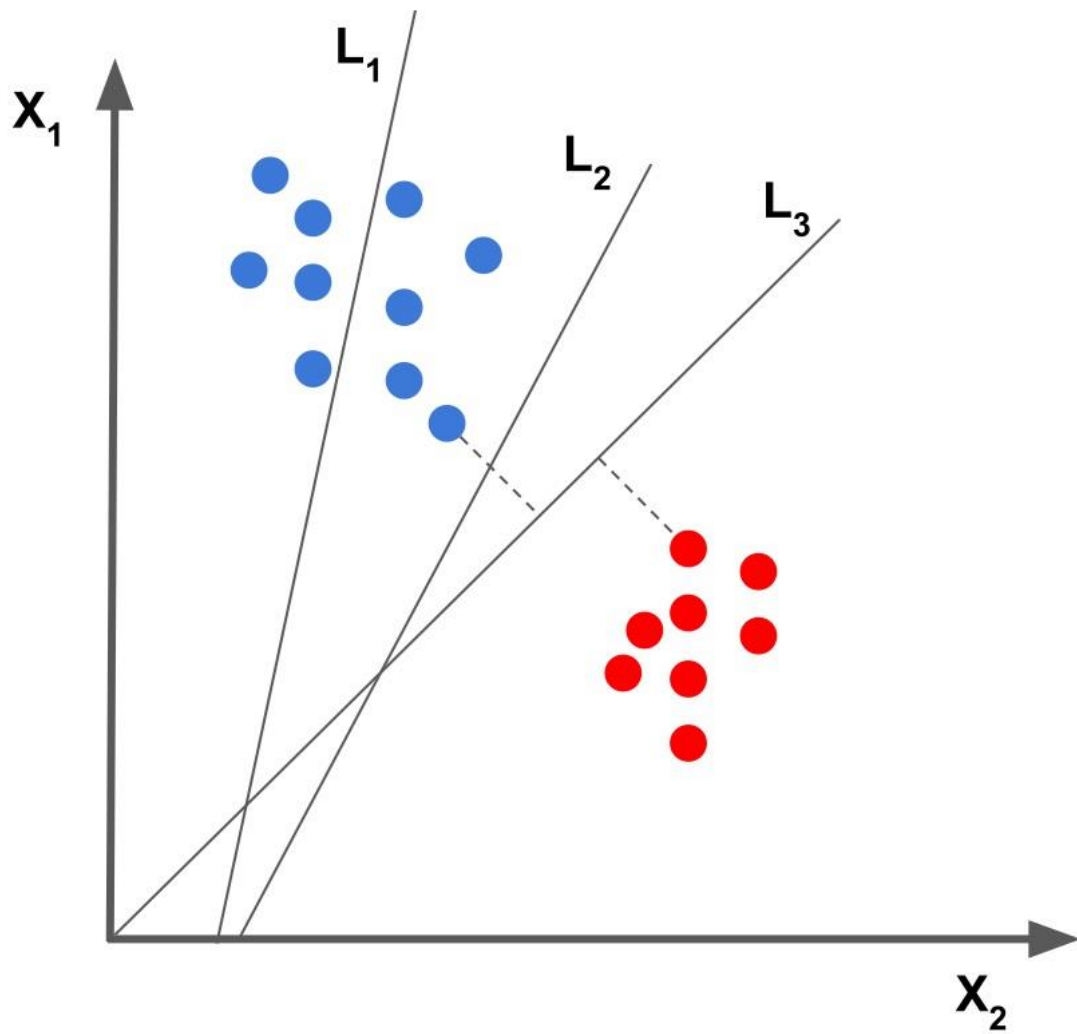
Generally, Yolo and SSD are preferred for real-time detection tasks, since their computational complexity is lower than Faster R-CNN. However, if accurate detection is important, which is the case for the tiger detection task, Faster R-CNN can be preferred. Generally Faster R-CNN performs better than Yolo and SSD in terms of accuracy. Therefore, Faster R-CNN is selected in this study for the task of tiger detection.

In this project, I would be focusing more on using the advantages of computer vision for the wild life conservation. There are a large number of methodologies used for computer vision. Some are better suited for some applications such as LBPH and PCA for facial recognition (Zhao & Wei, 2017; Kaur & Himanshi, 2015).

The most recurring methodologies that have come up in the application of machine learning for wild life conservation are SVM and CNN.

- ***Support Vector Machine (SVM)***

SVM is an approach to data classification that can also be used for linear regression. It basically involves separating data into classes, based on a distinguishing factor that is determined during training. Figure 2.2 shows a graphical representation of a classification process by SVM with three separating lines representing distinguishing factors. In this example, L3 is the best line.



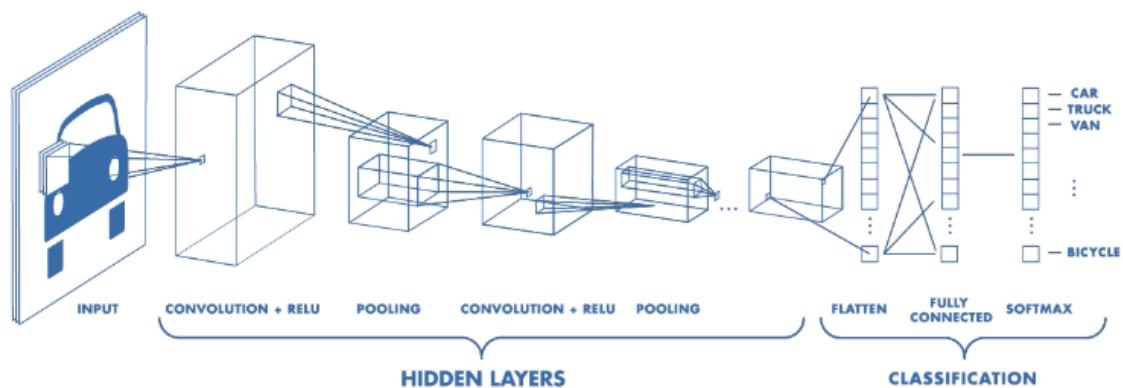
**Figure 2.2:** Graphical Representation of the Classification Process in SVM

SVM is a supervised learning algorithm and thus would need a dataset of annotated data. SVM has a low training time and good accuracy.

- **Convolutional Neural Networks (CNN)**

CNN is a class of deep learning networks that is known for its high accuracy and effectiveness. It's most popular use is for image and video classification and object detection. CNNs are made up of many multilayer perceptrons stacked up on each other but regularized with weight values to avoid overfitting. They are made up of an input layer, multiple hidden layers, and an output layer. The hidden layers are arranged sequentially such that the outputs of any particular node in the hidden layer is the input of the next layer. The hidden layer is made up of multiple convolution layers and pooling layers. With operations and activation functions acting on the inputs of each node.

The functionality of a CNN can be divided into two parts, for feature extraction and classification. In feature extraction, the neural network extracts important features from the input image to be used in the classification. The feature extraction part includes convolution layers to convolve the input data and pooling layers to down sample the input and avoid overfitting. The classification part involves a fully connected layer which is then connected to the output layer which has its own activation function that actually makes the final inference. Figure 2.3 shows a typical architecture of a CNN.



**Figure 2.3:** A typical architecture of a CNN

- Convolution layer

The convolution layer contains nodes that perform operations on the input to produce an output. The operations performed on the inputs of this layer are based on an activation function. In the general study of neural networks there are a number of activation functions that can be used in the convolutional layer. Some examples are sigmoid functions, tanh functions, ReLu, Leaky ReLu, etc.

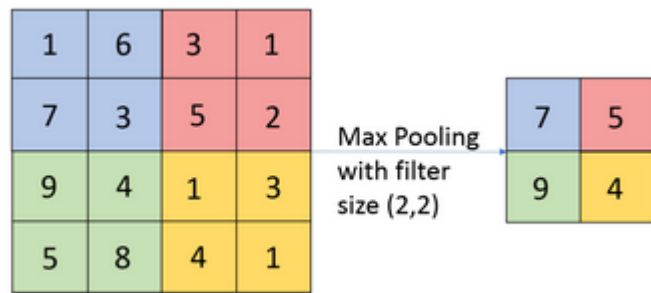
In this project, I would be using the ReLu activation function. This is a very simple linear activation function. In ReLu, positive inputs give positive outputs of the same value while negative inputs give zero as the output. Figure 2.4 shows the mathematical representation of the ReLu activation function.

$$ReLU(X) = \begin{cases} 0 & x < 0 \\ x & x \geq 0 \end{cases} = \max(0, x)$$

**Figure 2.4:** The ReLu activation function

- Pooling Layer

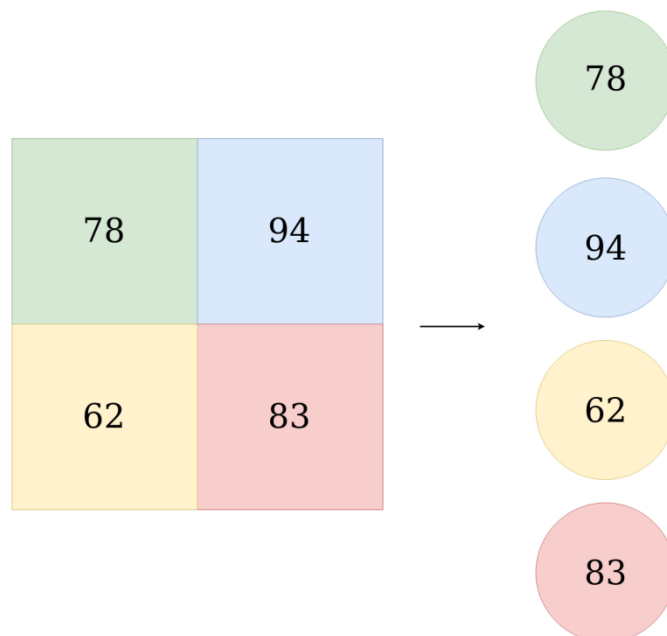
The pooling layer typically follows a convolutional layer and is supposed to downsize or summarize the output from the previous convolutional layer. The most common forms of pooling are average pooling and max pooling. Average pooling involves the calculation of the average value of each part of the input kernel. Max pooling is the most popular form of pooling, and it involves selecting the highest values from each part of the input kernel. Figure 2.5 shows the operation of max pooling.



**Figure 2.5** Max pooling with a 2x2 filter

- Fully Connected Layer

Here the output of the last pooling or convolutional layer (typically a 3-dimensional matrix) is unrolled or flattened into a vector. The fully connected layer is also sometimes referred to as the flattened layer. Figure 2.6 shows the flattening operation.



**Figure 2.6** The flattening process

- The Output Layer

This layer is where the final inference is made in the CNN. There can be only one output layer, and it consists of a single node for each class that the input can be sorted into. It also has its own type of activation function. The most popular activation function for output layers is Softmax which returns the classification and its probability distribution. The class with the highest probability distribution is the inferred class.

## CHAPTER 3

### DESIGN AND SOFTWARE TOOLS

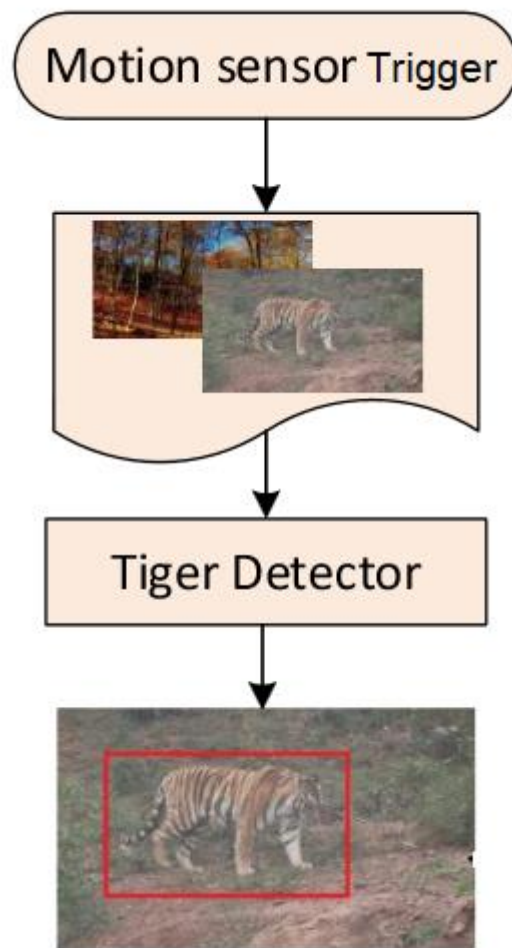
For the project, I first had to find the best software tools to serve the purpose. There is a large amount of tools, APIs and libraries available for work and research in the area of machine learning, but research has made it clear that specific approaches to machine learning serve better for specific applications. With this in mind, I had to make sure I found the best machine learning approach to serve my purpose and design my system accordingly. And in this aim I chose to use the Faster R-CNN which gives the best accurate within acceptable training time.

Which is the most important for the wildlife conservation aim.

#### 3.1 Design

The first objective of the system is to capture images of tigers in the wild. One viable approach would be to have an object detection module built into cameras that would be installed all over the tigers inhabit. This module would direct the cameras to capture anything that it considers to be a tiger in the wild. The challenge of this approach however is that it would require the cameras to be constantly on, which would cost more power and require more sophisticated power supplies.

The alternative to this approach - and the one employed in this project - is one where the cameras are fitted with motion sensors that capture any movement in the area the tigers inhabit and then feed it to the object detection module which would sort through the pictures for only pictures containing tigers. This way, the cameras are activated by motion sensors and do not need to be constantly switched on. Figure 3.1 shows a flowchart of the design of the system.



**Figure 3.1:** A flowchart of the system design

Considering the aim, it was clear that I needed an object detection system that would be able to identify individual tigers in an image. This way, the system would be able to specify if the image captured by the motion sensor camera at least contained a tiger or just another animal or object that was not the subject of the research. For this purpose, I needed a strong object detection mechanism. For this purpose, I chose to use the Faster R-CNN object detection architecture.



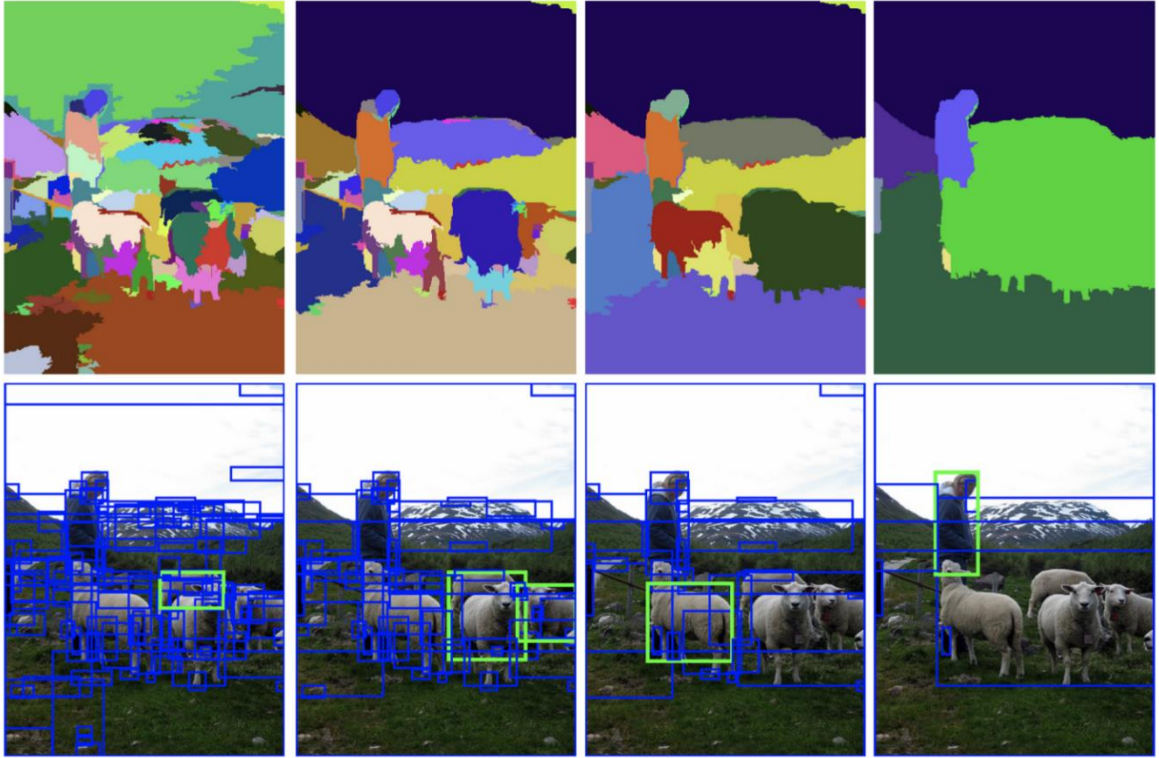
### 3.1.1 Faster R-CNN

Faster R-CNN is an object detection architecture that was designed by Shaoqing Ren, Kaiming He, Ross Girshick and Jian Sun. It was created as a solution to the complexity required for accurate localization in object identification problems and as an improvement on the initial R-CNN and Fast R-CNN.

Most object detection methods rely on a hypothesis, or guess about the location of the object to be detected. This led to the development of *region proposal* algorithms and the accuracy of most object detection approaches now rely greatly on the effectiveness of their region proposal. While this is so, it was later found that the region proposal algorithm could also make object identification very slow, because a CNN is applied to each and every one of the proposals. As a solution to this, the Faster R-CNN was developed (Ren et al; 2017), with a *Region Proposal Network* (RPN) that shares full-image convolutional features with the detection network.

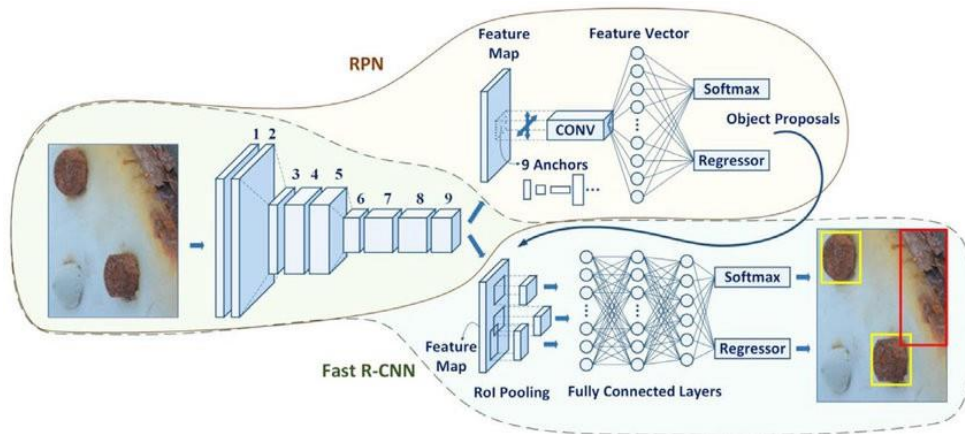
The Faster R-CNN is made up of two parts.

- A CNN that offers hypothetical locations of the object to be identified. This is the RPN. It tells the second part of the Faster R-CNN where it thinks it should look to find the object. The RPN takes an image as input and returns rectangular objects which are its proposals as the locations of the object. It also returns a value that represents its possibility of membership to a particular class as opposed to the rest of the image. Here, search selective is used to choose sections that have similar textures and colors and puts them in separate boxes. Softmax activation function is used to classify these selections and they are given as output. A linear regressor is also used to create bounding box localizations for each of the layers. This serves as the feature extraction process.



**Figure 3.2:** Classifications of an image into regions with similar textures and colours by the initial CNN. Some of these areas would be selected by the ROI pool to be passed to the fully connected layer

- The second part of the Faster R-CNN is the Fast R-CNN object detector. It takes the feature maps that are the output of the initial CNN and performs ROI pooling on them. ROI pooling extracts the regions of interest from the collection of regions given as output from the initial CNN. These ROI are then passed on to a fully connected layer where they are flattened and passed on to the output layer to be classified and given bounding boxes.



**Figure 3.3:** Faster R-CNN architecture

One of the key distinctions of the Faster R-CNN is that the whole system acts as a single network and shares convolutional layers between the RPN and the Faster R-CNN. This makes it unnecessary to train two separate networks.

In this project, I would be using TensorFlow's Faster R-CNN Inception V2 Model from TensorFlow's model zoo. The configuration presets for the model include, learning rate, epochs, validation period, batch size, GPU devices (or you could use a TPU), etc. The configuration parameters are available in the documentation.

### 3.2 Software Tools

As with any other neural network, I needed two major tools - a neural network framework or API and a dataset.

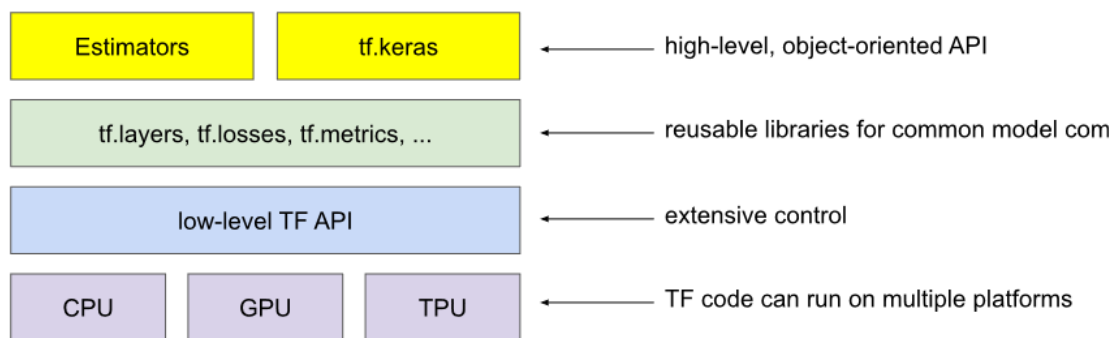
The neural network framework is essentially a platform with modules and functions to act as building blocks in the development of your neural network. There are different neural networks that work better for different objectives. Some of them are also made with support for specific programming languages.

The dataset consists of the data that the neural network is going to be trained on. This is a very important aspect of the development process, because the quality of the dataset affects the quality of the system. A dataset that is not robust enough could lead to a high amount of false positives in its application. Vijay D'Souza, the director of enhanced analytics at the United States Government Accountability Office talked about how it is important to understand the quality of data used in machine learning applications, because it determines how much we can rely on the data to make good decisions (7wData, 2017). Also, IBM estimates that bad data costs the US economy about \$3.1 trillion per year. This shows that using bad data in research could be dangerous, causing researchers to make bad assumptions and consequently, bad decisions.

### **3.2.1 TensorFlow**

TensorFlow is an open source platform for machine learning written in Python, C++ and CUDA. TensorFlow was initially designed by the Google Brain team for internal use at Google, but was later released under the Apache license in 2015. The can be used for a number of machine learning applications - most commonly, for the development of Neural Networks.

The TensorFlow APIs are stacked hierarchically, with the low level APIs used mainly to experiment with and develop new machine learning algorithms, and the high level APIs used to develop and train machine learning models. Figure 3.3 shows the arrangement of the TensorFlow APIs in the TensorFlow Toolkit.



**Figure 3.4:** The TensorFlow API Hierarchy

Most of the interactions for the purpose of this project would be on the high level object oriented APIs.

The reusable libraries consist of modules that are predefined in the framework and can be used across machine learning applications. They can be used to define layers, metrics, etc, in the development of your machine learning model. The `tf.keras` API provides extensibility that allows the development of new libraries if needed.

The low level API is a looser level that allows the definition of all core variables, layers, parameters and even definition of placeholders. This level gives more control over the structure of the machine learning model and is more complicated. This level requires a higher level of expertise to use.

The last level is the hardware level which consists of the processing unit to be used in the learning and actual application of the machine learning model. Most developers use GPUs to allow for more processing power, and Google also created the TPU which is an application-specific Integrated circuit for neural network machine learning done using Tensor Flow.

- ***High level Object-Oriented API***

This level consists of the Estimators and Keras. The Estimators (`tf.estimator`) encapsulates the functions:

- Training
- Evaluation
- Prediction
- Export for serving

The Tensor flow framework allows you to use the predefined estimator or create your own custom estimator although they would be based on the `tf.estimator.Estimator` class. An estimator is simply a model-level abstraction that performs the basic core functions of a machine learning model. Estimators help to easily control how you load data, handle exceptions, create checkpoint files and recover from system failures. Estimators are no longer used as much as before, and are no longer covered in the Google developers machine learning crash course, although some of its capabilities, such as parameter server based training and full TFX integration are under development for `tf.keras`.

`tf.keras` is a TensorFlow high level API that is used for fast prototyping, high-end research and production. Its major advantages are that it is:

- User-friendly
- Modular
- Easy to extend

Keras has a simple user interface designed for common use cases and it provides clear feedback of user errors. Also, its models are made up of configurable components that can be combined easily with few restrictions. It also allows you to write and develop custom new components, layers, metrics, etc which helps to easily express new ideas in research efforts.

### 3.2.2 The ATRW Dataset

As has become clear over the years of research in machine learning, the dataset used to train a machine learning algorithm is a very important part of any work to be done in machine learning. Therefore, it is important that I discuss the dataset used in this project.

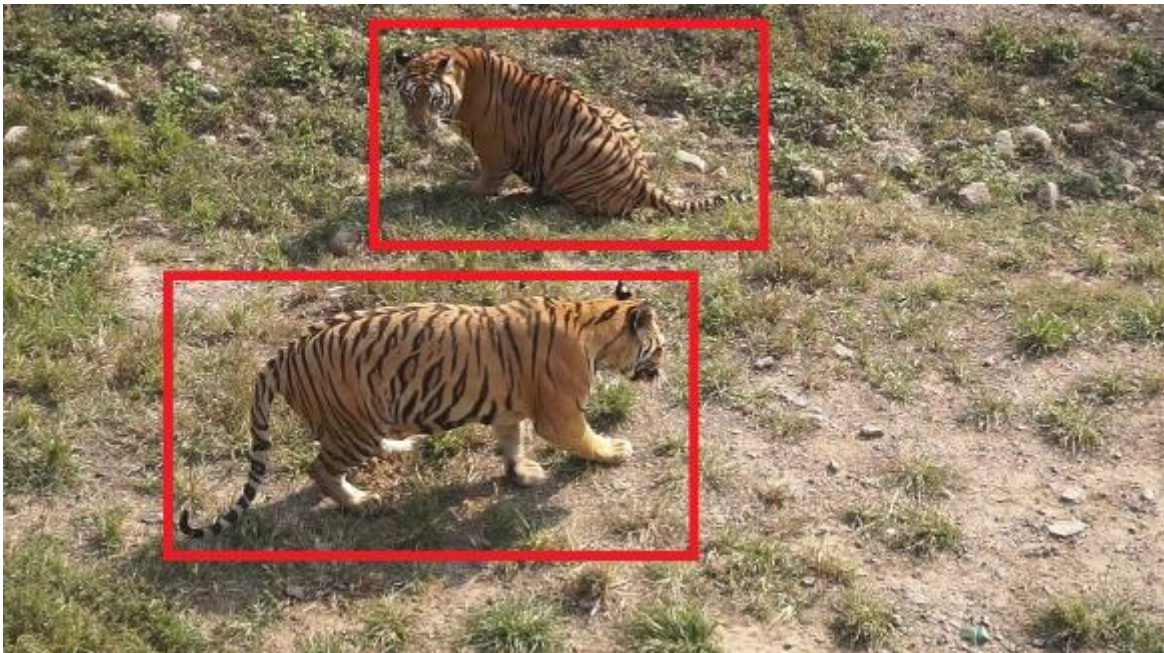
The dataset I used in this project was provided under the CC BY-NC-SA 4.0 license - which is for the purpose of non-commercial research - on the CVWC (Computer Vision for Wildlife Conservation) challenge website. They processed sample video frames from 8,000 video clips of 92 Amur tigers taken from 10 zoos in china by a third party company called *MakerCollider*, with the help of the Worldwide Fund for Nature (WWF). The dataset is known as the ATRW (Amur Tiger Re-identification in the Wild) dataset (Li et al, 2019) and was created along with its annotations with a focus on aiding re-Identification for wildlife and is available with bounding boxes, pose key points, and tiger identity annotations. The dataset contains 2,760 images of size 1920x1080. I divided it into two groups of 2,060 images for training and 700 images for testing. This makes a 3-to-1 ratio or 75% to 25% division.

Most re-Identification methods that have been created recently have been designed for urban settings with focus on pedestrians and cars. This therefore makes it impractical to use the same re-Identification methods for wildlife because pedestrians and cars have less pose variations as opposed to wildlife. Also, backgrounds in the wild are more complex than in an urban setting, making re-Identification more challenging. This is why making a dataset with a focus on re-Identification in the wild was very necessary.

- ***Dataset Annotation***

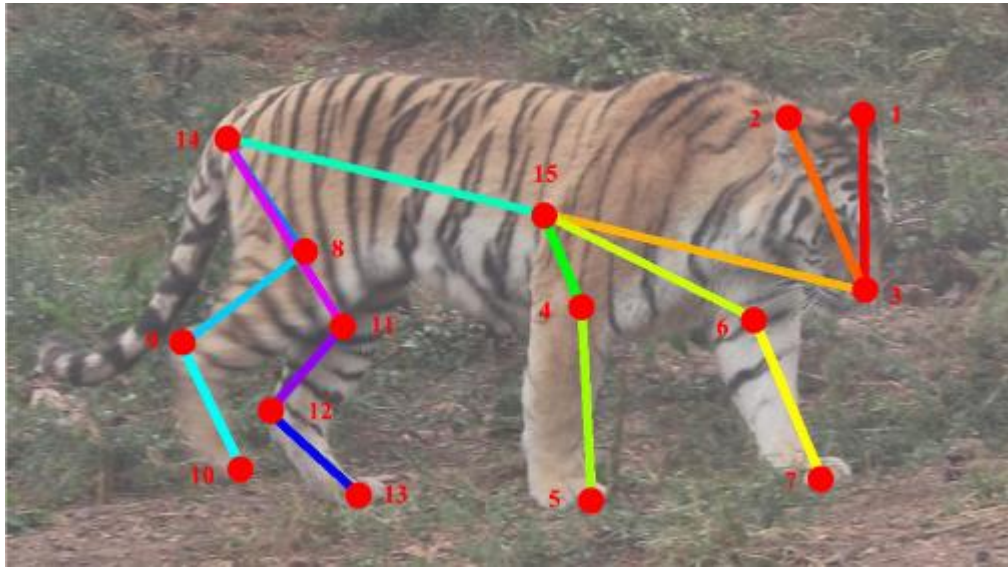
The annotation of the dataset was done in three steps.

- A bounding box is assigned for each tiger, as well as the view of the tiger, to specify if it is facing the front, back, left or right. Figure 3.4 shows the tigers within their bounding boxes.
- Professionals define the tiger identity, based on appearance. Tigers from different zoos were separated during the annotation to make sure the annotation is as accurate as possible. Any tigers that could not be identified was given an unknown identity.
- As a four legged animal, the tigers movements caused a wide range of pose variations. Therefore, skeleton key points were annotated for each tiger for downstream pose normalization. Figure 3.5 shows a tiger with its annotated skeletal key points.



**Figure 3.5:** Tigers within their bounding boxes





**Figure 3.6:** A tiger with its annotated skeletal key points

## CHAPTER 4

### METHODOLOGY AND DEVELOPMENT

After creating the design of the system and selecting the best tools for my objective, I moved on to the development process. First of all, I did all of my work on Google Colab because it allows me to execute the code on Google Clouds server, and take advantage of Google hardware, such as GPUs and TPUs. This way, I could use TensorFlow GPU, rather than the regular TensorFlow. TensorFlow GPU provides more processing power and reduces the training time of the neural network by a factor of eight, so this was very helpful.

The development process was done in a number of steps which would be discussed in this chapter.

#### 4.1 Data Acquisition

First of all, I downloaded the ATRW dataset consisting of the image dataset and its corresponding annotation files in XML format from the CVCW website. Next, I split the dataset into two parts for training and testing to a 75:25 ratio. The commands in Figure 4.1 show the process.

```
$ mkdir train
$ mkdir val
$ while IFS= read -r filename; do mv "$filename".xml train/; done < train.txt
$ while IFS= read -r filename; do mv "$filename".jpg train/; done < train.txt
$ while IFS= read -r filename; do mv "$filename".xml val/; done < val.txt
$ while IFS= read -r filename; do mv "$filename".jpg val/; done < val.txt
```

**Figure 4.1:** Splitting the dataset

## 4.2 Set Up Working Environment

At this stage, I needed to set up my working environment. I needed to install some software packages, libraries, and dependencies. I also needed to define my directory structure to allow me to easily navigate through the folders.

Firstly, I created a folder called *tensorflow1* to contain my all files and the datasets. I then downloaded the TensorFlow object detection Library from github. I extracted it into my *tensorflow1* folder and named it as *models*.

Next, I downloaded the Faster-RCNN-Inception from TensorFlow's model zoo, which is where TensorFlow's object detection models are located. The commands in Figure 4.2 are used to download and simultaneously extract Faster RCNN Inception.

```
$ wget http://download.tensorflow.org/models/object_detection/faster_rcnn_inception_v2_coco_2018_01_28.tar.gz
$ tar -xf faster_rcnn_inception_v2_coco_2018_01_28.tar.gz
```

**Figure 4.2:** Downloading and extracting Faster RCNN Inception

After these processes, I end up with the folder structure shown in Figure 4.3.

```
tensorflow1
|
|-- models
|   |-- research
|       |-- object_detection
|           |-- images
|               |-- train
|               |-- val
|               |-- faster_rcnn_inception_v2_coco_2018_01_28
|
```

**Figure 4.3:** My folder structure

Next, I needed to configure my PYTHONPATH environment variables and install my dependencies. Pandas and open-cv packages were not actually necessary with TensorFlow, However, I needed them for the python scripts I would use to generate TFRecords files to use as input in the TensorFlow model. Figure 4.4 shows this configuration and installation process.

```
$ export PYTHONPATH=/home/dgxuser104/Balmukund/tensorflow1/models:/home/dgxuser104/Balmukund/tensorflow1/models/research:/home/dgxuser104/Balmukund/tensorflow1/models/research/slim:/home/dgxuser104/Balmukund/tensorflow1/models/research/object_detection
$ export PATH=$PATH:$PYTHONPATH
$ pip install tensorflow-gpu==1.8
$ pip install pillow lxml Cython matplotlib pandas opencv-python
```

**Figure 4.4:** Configuring PYTHONPATH environment variables and installing dependencies

Next, I needed to compile the protobuf files, for the configuration of all the parameters that TensorFlow uses for modelling and training. To do this, I navigated to the research directory and pasted the code shown in Figure 4.5.

```
$ protoc --python_out=object_detection object_detection/protos/  
anchor_generator.proto object_detection/protos/argmax_matcher.proto  
object_detection/protos object_detection/protos/anchor_generator.proto  
object_detection/protos/input_reader.proto object_detection/protos/  
losses.proto object_detection/protos/matcher.proto object_detection/  
protos/argmax_matcher.proto object_detection/protos/bipartite_matcher.  
proto object_detection/protos/box_coder.proto object_detection/protos/  
box_predictor.proto object_detection/protos/eval.proto object_detection/  
protos/faster_rcnn.proto object_detection/protos/faster_rcnn_box_coder.  
proto object_detection/protos/grid_anchor_generator.proto  
object_detection/protos/hyperparams.proto object_detection/protos/  
image_resizer.proto object_detection/protos/input_reader.proto  
object_detection/protos/losses.proto object_detection/protos/matcher.  
proto object_detection/protos/mean_stddev_box_coder.proto  
object_detection/protos/model.proto object_detection/protos/optimizer.  
proto object_detection/protos/pipeline.proto object_detection/protos/  
post_processing.proto object_detection/protos/preprocessor.proto  
object_detection/protos/region_similarity_calculator.proto  
object_detection/protos/square_box_coder.proto object_detection/protos/  
ssd.proto object_detection/protos/ssd_anchor_generator.proto  
object_detection/protos/string_int_label_map.proto object_detection/  
protos/train.proto object_detection/protos/keypoint_box_coder.proto  
object_detection/protos/multiscale_anchor_generator.proto  
object_detection/protos/graph_rewriter.proto object_detection/protos/  
calibration.proto object_detection/protos/flexible_grid_anchor_generator.  
proto
```

**Figure 4.5:** Compiling the protobuf files

Finally, we compile `setup.py` as seen in Figure 4.6.

```
$ python setup.py build  
$ python setup.py install
```

**Figure 4.6:** Compiling `setup.py`

### 4.3 Data preprocessing

For the next step, I needed to generate a tfrecords file from the dataset to serve as the input data for my TensorFlow neural network model. To do this, I used a python script to first of all convert the XML data to CSV, then another script to convert it from CSV to TFRecords. The code used for these two conversions would be added in the appendix section. The file paths are modified within the code to convert both the training and testing sets.

```
$ python xml_to_csv.py
```

**Figure 4.7:** Command to convert the XML data to CSV

This would create a files named “train\_labels.csv” and “val\_labels.csv” in the images folder.

Then I opened the `generate_tfrecord.py` file and set the label to “Tiger” as shown in Figure 4.8.

```
def class_text_to_int(row_label):  
    if row_label == 'Tiger':  
        return 1  
    else:  
        return None
```

**Figure 4.8:** Definition of the label map

Next, I had to create the label map which identifies objects to the training model by mapping names to IDs. To do this, I created a new file called `labelmap.pbtxt` in the `training_faster_rcnn` folder, under the `object_detection` folder. Figure 4.9 shows this mapping.

```
item {
  id: 1
  name: 'Tiger'
}
```

**Figure 4.9:** The label map

#### 4.4 Training Configuration

Here, we configure our training pipeline. First, I navigate to *object\_detection/samples/configs* and copy *faster\_rcnn\_inception\_v2\_pets.config* into */object\_detection/training\_faster\_rcnn*, open the file, and make the following changes.

- Set `num_classes` to 1 since we only have one class to be identified, which is the tiger.
- Set `fine_tune_checkpoint` to:  
"tensorflow1/models/research/object\_detection/faster\_rcnn\_inception\_v2\_coco\_2018\_01\_28/model.ckpt"
- In the `train_input_reader` section, set `input_path` to:  
"tensorflow1/models/research/object\_detection/images/train.record". Also set `label_map_path` to:  
"tensorflow1/models/research/object\_detection/training\_faster\_rcnn/labelmap.pbtxt"
- Set `num_examples` to the number of images in `images/test`. In this case, that would be set to 443.
- In the `eval_input_reader` section, set `input_path` and `label_map_path` to  
"tensorflow1/models/research/object\_detection/images/val.record" and  
"tensorflow1/models/research/object\_detection/training\_faster\_rcnn/labelmap.pbtxt" respectively.

After this, I save the file and train the model.

## 4.5 Training the Model

Here, I run the command shown in Figure 4.10 from the `object_detection` folder.

```
$ CUDA_VISIBLE_DEVICES=7 python train.py --logtostderr
--train_dir=training_faster_rcnn/
--pipeline_config_path=training_faster_rcnn/
faster_rcnn_inception_v2_pets.config
```

**Figure 4.10:** Training the neural network model

If the setup is correct, TensorFlow would take 30 seconds to initialize before it starts training. Each step reports a loss which goes down as the training goes on.

## 4.6 Saving and Inference

After the training process, I needed to generate an inference graph of the model. To do this, I run the command in Figure 4.11 from the `object_detection` folder.

---

```
$ CUDA_VISIBLE_DEVICES=7 python export_inference_graph.py --input_type
image_tensor --pipeline_config_path training_faster_rcnn/
faster_rcnn_inception_v2_pets.config --trained_checkpoint_prefix
training_faster_rcnn/model.ckpt-XXXX --output_directory /
training_faster_rcnn/inference_graph
```

**Figure 4.11:** Command to generate the inference graph file

This would create a file called `frozen_inference_graph.pb` in the `/object_detection/training_faster_rcnn/inference_graph` folder which would contain the object detection classifier.



## CHAPTER 5

### RESULTS AND DISCUSSION

The training process lasted for about 7 hours and went smoothly. This was a good thing because, it would have run for close to 24 hours on the regular TensorFlow, as opposed to the 7 hours I got on the TensorFlow-GPU API. Also, the framework saved checkpoints at intervals of about 5 minutes. The loss value started at 3.0 and went as low as 0.106 over 200,000 steps.

The Faster R-CNN model is a novel approach to neural networks for object detection that has clearly succeeded in maintaining - and in some cases, improving - the high level of accuracy of the normal Convolutional Neural Networks while also reducing the time spent in training. The Faster R-CNN is simply an improvement on the Fast R-CNN which in turn is a step forward from the CNN.

#### 5.1 Tests and Comparison

To show the superiority of the Faster R-CNN model, Table 5.1 shows a comparison between it and the SSD inception V2 and the SSD Lite MobileNet, which are also models from the TensorFlow model zoo that were used to develop Object identification systems for the CVWC challenge (Li et al, 2019).

**Table 5.1:** Comparison of performances for Faster R-CNN, SSD and SSD Lite MobileNet

	<b>Faster R-CNN</b>	<b>SSD</b>	<b>SSD Lite MobileNet</b>
<i>mAP at IoU</i>	0.608	0.476	0.426
<i>Steps</i>	250,000	100,000	130,000

mAP is the mean average precision. The formula for precision is given below in Figure 5.1. IoU means the intersection over the union which is the overlap between the ground truth and the prediction. This needs to be above 0.5 for a good prediction. The formula for IoU is given in Figure 5.2.

$$\text{Precision} = \frac{\text{tp}}{\text{total positive results}}$$

**Equation 5.1:** Formula for precision

$$\text{IoU} = \frac{\text{area of overlap}}{\text{area of union}}$$

**Equation 5.2:** Formula for IoU

The Faster R-CNN model performed better with higher mAP. Results for the SSD and SSD MobileNet are received from the CVWC2019 challenge (Li et. al, 2019). In cases where there is available computation power, the Faster R-CNN model would be the better choice in terms of accuracy. The SSD Lite might be the preference in a case where the system would be run on an embedded device, such as a Raspberry Pi. But if the computation is

performed in the Cloud using Raspberry Pi and Python, then Faster R-CNN can be preferred as well.

## **5.2 Constraints and Limitations of the study**

During the development process, a few issues were encountered along the way that either slowed down the development process, or made it necessary to find alternative methods of approach.

One of the issues was with the dataset. The data to be used in TensorFlow should be in the TFRecords format, but the dataset was in XML format. For this reason, I had to first find a script to convert the dataset to CSV and then another script to convert it to TFRecords.

Also, the TensorFlow API requires that you maintain the directory structure in their Github repository, so I had to keep that as it was. They also required a number of software tools and dependencies which I had to set up before moving forward.

Finally, in the compilation of the Protobuf files, the protoc compilation command given in the TensorFlow object detection API installation instructions did not work. As a result, I had to call out all the .proto files in the protos folder individually.

## CHAPTER 6

### CONCLUSION AND FUTURE WORKS

#### 6.1 Conclusion

For a system like the one I am trying to develop, the Faster R-CNN would be the better option because the computation would not be taking place on the camera or on any embedded computers. This way, the images are captured by the motion sensor cameras and fed to the neural network model. The images containing tigers are the selected by the model and made available to wildlife conservationists to analyse and strategize on the best ways to save the endangered species.

#### 6.2 Future Works

The Faster R-CNN is a new approach to tiger detection. As with all other models that have been designed, and its predecessor, the Fast R-CNN, it can be improved to give even better results.

As most machine learning models have started to get higher and higher levels of accuracy making machine learning more reliable, we can now expand the scope of the application of machine learning in all these fields. In wildlife conservation, It can be applied to also monitor and study animal movement and migration patterns. It can be used to see how changes in environment and other factors affect the animal life and wellbeing. It could also in some cases be used to run predictive models to see how certain changes can affect the wildlife in an area, thus helping industries to set boundaries on their operations and avoid harming the wildlife.

We can improve our system to detect more classes such as (elephants, lions and other types) if we can get datasets for these animals, also these information could be used to monitor and analyze these endangered species life and investigate the best ways to protect them.

The applications for machine learning in wildlife conservation go beyond monitoring endangered species and their habitat, but can also be used for analysis of big data to avoid endangering any more species.

## REFERENCE

- 7wData. (2018, May 7). *Bad Data Is Ruining Machine Learning, Here's How To Fix It*. Retrieved May 25, 2020, from <https://www.7wdata.be/data-management/bad-data>
- Attal, Z. and C. Direkoglu (2019). *Sea Turtle Species Classification for Environmental Research and Conservation*. International Conference on Theory and Application of Soft Computing (ICSCCW 2019). Advances in Intelligent Systems and Computing (LNCS), vol 1095, 580-587, 2019.
- Badawy, M. and C. Direkoglu (2019). *Sea Turtle Detection Using Faster R-CNN for Conservation Purpose*. International Conference on Theory and Application of Soft Computing (ICSCCW 2019). Advances in Intelligent Systems and Computing (LNCS), vol 1095, 535-541, 2019.
- Chen, G., Han, T.X., He, Z., Kays, R., & Forrester, T. (2014). *Deep convolutional neural network based species recognition for wild animal monitoring*. 2014 IEEE International Conference on Image Processing (ICIP), 858-862.
- Extracting business value from the 4 V's of big data*. (n.d.). Retrieved May 25, 2020, from <https://www.ibmbigdatahub.com/infographic/extracting-business-value-4-vs-big-data>
- Fang, F., Nguyen, T.H., Pickles, R., Lam, W.Y., Clements, G.R., An, B., Singh, A., Tambe, M., & Lemieux, A. (2016). *Deploying PAWS: Field Optimization of the Protection Assistant for Wildlife Security*. AAAI.
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). *Deep Residual Learning for Image Recognition*. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770- 778.
- Hogeback, J. (n.d.). *What Makes a Species Endangered?* Retrieved May 12, 2020, from <https://www.britannica.com/story/what-makes-a-species-endangered-is-ruining-machine-learning-heres-how-to-fix-it/>
- Kaur, R., & Himanshi, E. (2015). *Face recognition using Principal Component Analysis*. 2015 IEEE International Advance Computing Conference (IACC). doi:10.1109/iadcc.2015.7154774

- Khazri, A. (2019, April 9). *Faster RCNN Object detection*. Retrieved May 17, 2020, from <https://towardsdatascience.com/faster-rcnn-object-detection-f865e5ed7fc4>
- Levy, E. (n.d.). *Are Tigers Endangered? What Does Endangered Even Mean?* •Earth.com. Retrieved May 12, 2020, from <https://www.earth.com/earthpedia-articles/are-tigers-endangered/>
- Li, S., Li, J., Lin, W., & Tang, H. (2019). *Amur tiger re-identification in the wild*. arXiv preprint arXiv:1906.05586.
- Margalef, R. (1968). *Perspectives in ecological theory*. Chicago: University of Chicago Press.
- Norouzzadeh, M.S., Nguyen, A., Kosmala, M., Swanson, A., Packer, C., & Clune, J. (2017). *Automatically identifying wild animals in camera trap images with deep learning*. ArXiv, abs/1703.05830.
- Ren, S., He, K., Girshick, R., & Sun, J. (2017). *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6), 1137–1149. doi: 10.1109/tpami.2016.2577031
- Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... Fei-Fei, L. (2015). *ImageNet Large Scale Visual Recognition Challenge*. *International Journal of Computer Vision*, 115(3), 211–252. doi: 10.1007/s11263-015-0816-y
- Singh, V. P., & Odaki, K. (2004). *Mangrove ecosystem: structure and function*. Jodhpur: Scientific Publishers (India).
- Thangarasu, R., Kaliappan, V. K., Surendran, R., Sellamuthu, K., & Palanisamy, J. (2019). *Recognition Of Animal Species On Camera Trap Images Using Machine Learning And Deep Learning Models*. *International Journal of Scientific & Technology Research*, 8(10), 2613–2622.
- Usher M.B. (1986) *Wildlife conservation evaluation: attributes, criteria and values*. In: Usher M.B. (eds) *Wildlife Conservation Evaluation*. Springer, Dordrecht
- Usher, M. B. (1973). *Biological Management and Conservation* (1st ed.). Springer US. doi: 10.1007/978-1-4899-3410-9
- Zhao, X., & Wei, C. (2017). *A real-time face recognition system based on the improved LBPH algorithm*. *2017 IEEE 2nd International Conference on Signal and Image Processing (ICSIP)*. doi:10.1109/siprocess.2017.8124508

## **APPENDICES**



## APPENDIX 1

### XML to CSV Converter

```
import os
import glob
import pandas as pd
import xml.etree.ElementTree as ET

def xml_to_csv(path):
    xml_list = []
    for xml_file in glob.glob(path + '/*.xml'):
        tree = ET.parse(xml_file)
        root = tree.getroot()
        for member in root.findall('object'):
            value = (root.find('filename').text,
                    int(root.find('size')[0].text),
                    int(root.find('size')[1].text),
                    member[0].text,
                    int(member[4][0].text),
                    int(member[4][1].text),
                    int(member[4][2].text),
                    int(member[4][3].text)
                    )
            xml_list.append(value)
    column_name = ['filename', 'width', 'height', 'class',
                  'xmin', 'ymin', 'xmax', 'ymax']
    xml_df = pd.DataFrame(xml_list, columns=column_name)
    return xml_df
```

```
def main():  
    image_path = os.path.join(os.getcwd(), 'annotations')  
    xml_df = xml_to_csv(image_path)  
    xml_df.to_csv('raccoon_labels.csv', index=None)  
    print('Successfully converted xml to csv.')
```

```
main()
```

## APPENDIX 2

### Code to Generate TFRecord from CSV (generate\_tfrecord.py)

```
"""
Usage:
  # From tensorflow/models/
  # Create train data:
  python generate_tfrecord.py --
csv_input=data/train_labels.csv --output_path=train.record
  # Create test data:
  python generate_tfrecord.py --
csv_input=data/test_labels.csv --output_path=test.record
"""

from __future__ import division
from __future__ import print_function
from __future__ import absolute_import

import os
import io
import pandas as pd
import tensorflow as tf

from PIL import Image
from object_detection.utils import dataset_util
from collections import namedtuple, OrderedDict

flags = tf.app.flags
flags.DEFINE_string('csv_input', '', 'Path to the CSV input')
flags.DEFINE_string('output_path', '', 'Path to output
TFRecord')
flags.DEFINE_string('image_dir', '', 'Path to images')
```

```

FLAGS = flags.FLAGS

# TO-DO replace this with label map
def class_text_to_int(row_label):
    if row_label == 'raccoon':
        return 1
    else:
        None

def split(df, group):
    data = namedtuple('data', ['filename', 'object'])
    gb = df.groupby(group)
    return [data(filename, gb.get_group(x)) for filename, x
in zip(gb.groups.keys(), gb.groups)]

def create_tf_example(group, path):
    with tf.gfile.GFile(os.path.join(path,
'{}'.format(group.filename)), 'rb') as fid:
        encoded_jpg = fid.read()
        encoded_jpg_io = io.BytesIO(encoded_jpg)
        image = Image.open(encoded_jpg_io)
        width, height = image.size

        filename = group.filename.encode('utf8')
        image_format = b'jpg'
        xmin = []
        xmax = []
        ymin = []
        ymax = []

```

```

classes_text = []
classes = []

for index, row in group.object.iterrows():
    xmin = row['xmin'] / width
    xmax = row['xmax'] / width
    ymin = row['ymin'] / height
    ymax = row['ymax'] / height
    classes_text.append(row['class'].encode('utf8'))
    classes.append(class_text_to_int(row['class']))

tf_example =
tf.train.Example(features=tf.train.Features(feature={
    'image/height': dataset_util.int64_feature(height),
    'image/width': dataset_util.int64_feature(width),
    'image/filename':
dataset_util.bytes_feature(filename),
    'image/source_id':
dataset_util.bytes_feature(filename),
    'image/encoded':
dataset_util.bytes_feature(encoded_jpg),
    'image/format':
dataset_util.bytes_feature(image_format),
    'image/object/bbox/xmin':
dataset_util.float_list_feature(xmins),
    'image/object/bbox/xmax':
dataset_util.float_list_feature(xmaxs),
    'image/object/bbox/ymin':
dataset_util.float_list_feature(ymins),
    'image/object/bbox/ymax':
dataset_util.float_list_feature(ymaxs),

```

```

        'image/object/class/text':
dataset_util.bytes_list_feature(classes_text),
        'image/object/class/label':
dataset_util.int64_list_feature(classes),
    )))
    return tf_example

def main(_):
    writer = tf.python_io.TFRecordWriter(FLAGS.output_path)
    path = os.path.join(FLAGS.image_dir)
    examples = pd.read_csv(FLAGS.csv_input)
    grouped = split(examples, 'filename')
    for group in grouped:
        tf_example = create_tf_example(group, path)
        writer.write(tf_example.SerializeToString())

    writer.close()
    output_path = os.path.join(os.getcwd(),
FLAGS.output_path)
    print('Successfully created the TFRecords:
{}'.format(output_path))

if __name__ == '__main__':
    tf.app.run()

```

## APPENDIX 3

### Similarity Report

Chapters	Percentages
Abstract.doc/docx	0%
Chapter 1.doc/docx	1%
Chapter 2.doc/docx	3%
Chapter 3.doc/docx	7%
Chapter 4.doc/docx	10%
Chapter 5.doc/docx	8%
Chapter 6.doc/docx	0%
*All.doc/docx	<=6%

\*All.doc/docx document must include all your thesis chapters (except cover page, table of contents, acknowledge, declaration, references, appendix, list of figures, list of tables, and abbreviations list ).

The screenshot shows a Turnitin submission page for a thesis. The page title is "Thesis" and it shows a list of submitted papers. A red box highlights the first seven rows of the table, which correspond to the data in the table above. The table columns are: AUTHOR, TITLE, SIMILARITY, GRADE, RESPONSE, FILE, PAPER ID, and DATE. The highlighted rows are for Ziad A, with titles ranging from "Overall Thesis Final" to "Chapter 1".

AUTHOR	TITLE	SIMILARITY	GRADE	RESPONSE	FILE	PAPER ID	DATE
Ziad A	Overall Thesis Final	6%	--	--		1334353155	29-May-2020
Ziad A	Chapter 4	10%	--	--		1334352642	29-May-2020
Ziad A	abstract	0%	--	--		1334352313	29-May-2020
Ziad A	Chapter 6	0%	--	--		1334031305	29-May-2020
Ziad A	Chapter 5	8%	--	--		1334031082	29-May-2020
Ziad A	Chapter 3	7%	--	--		1334030611	29-May-2020
Ziad A	chapter 2	3%	--	--		1334030202	29-May-2020
Ziad A	Chapter 1	1%	--	--		1334029753	29-May-2020
Manal D	chapter 4	90%	--	--		1332697183	27-May-2020
Manal D	chapter 4	90%	--	--		1332187713	26-May-2020
Manal D	chapter3	44%	--	--		1332119295	26-May-2020
Manal D	chapter2	96%	--	--		1332117590	26-May-2020
Manal D	chapter1	84%	--	--		1332117351	26-May-2020
Manal D	chp3	34%	--	--		1329886335	22-May-2020
Manal D	chapt 1and 2	35%	--	--		1329885998	22-May-2020
Ziad A	Thesis	11%	--	--		1326888278	18-May-2020

Regards,

Assoc. Prof Dr Melike Sah Direkoglu.

**APPENDIX 4**  
**Ethics Approval**

**ETHICAL APPROVAL DOCUMENT**

Date:   30   /   6   / 2020

To the **Graduate School of Applied Sciences**

The research project titled “...Efficient Model for Tiger Detection using Faster R-CNN.....” has been evaluated. Since the researcher(s) will not collect primary data from humans, animals, plants or earth, this project does not need to go through the ethics committee.

**Title: Assoc Prof Dr**

**Name Surname: Melike Sah Direkoglu**

**Signature:**



**Role in the Research Project: Supervisor**