PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)
100-ft Ring Road, Bengaluru – 560 085, Karnataka, India

A Project Report
On

Digital Borders: Animal Tracking through Re-identification

Submitted in fulfillment of the requirements for the Project phase -1

Submitted by

Prashanth C Ravoor
SRN: PES1201802670

Under the guidance of
Dr. Sudarshan T S B
Dean of Research, Professor

August- December 2019
FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

PROGRAM M.TECH

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

PROGRAM M.TECH

CERTIFICATE

This is to certify that the Dissertation entitled

'Digital Borders: Animal Tracking through Re-identification'

is a bonafide work carried out by

Prashanth C Ravoor
SRN: PES1201802670

In partial fulfillment for the completion of 3rd semester course work in the Program of Study MTECH in Computer Science and Engineering under rules and regulations of PES University, Bengaluru during the period Aug. 2019 – Dec. 2019. It is certified that all corrections/suggestions indicated for internal assessment have been incorporated in the report. The project report has been approved as it satisfies the 3rd semester academic requirements in respect of project work.

Signature with date & Seal	*Signature with date & Seal*	*Signature with date & Seal*
Internal Guide	*Chairperson*	*Dean of Faculty*

Name and Signatures of the Examiners

1.
2.
3.

TABLE OF CONTENTS

TABLE OF FIGURES

LIST OF TABLES

1 INTRODUCTION

Tracking objects in video sequences is not a new problem – it has been an active field of study for several years, and is primarily used for video surveillance of selected targets. The benefits of visual object tracking are well understood – the most common being security and surveillance, and even traffic control [1]. The essential function of object tracking is to trace the movement of a particular entity across a sequence of consecutive image frames, the result being a trajectory or a sequence of points representing the motion of the entity being tracked. Points obtained from recent frames can be used to predict the movement of the object over the next few image frames. This can be especially useful in continuously tracking the object when it is temporarily blocked from view by a different object in the foreground (occlusions). There can be a single object being tracked in a video sequence (single object tracking), or several (multi-object tracking).

Apart from occlusions, visual object tracking faces several challenges. To name a few:

a) The object being tracked can move out of the focal length of the camera, or move faster than the capture rate of the camera, turning it hazy (motion blur).
b) There could be different lighting conditions in different parts of the same video sequence, resulting in different illumination levels for the object at different points in time.
c) The object could turn in-place to provide a different orientation to the camera.
d) The object could entirely disappear from view (total occlusion) temporarily.
e) Two objects moving close to each other could be very similar in appearance, confusing the trajectory of the original object being tracked
f) The object could move into a part of the image where the background is visually similar to the foreground object (camouflage effect)
g) The object could move into the Field of Vision (FoV) of a different camera, which has different visual receptors, causing the appearance of the object to change slightly

The advent of deep neural networks has pioneered nearly every field in Computer Vision, and object tracking is no exception. Classical methods used for tracking, such as contour based tracking [2], Meanshift tracking [3], visual similarity based tracking such as color histograms [4] all perform poorly compared to neural networks. Kalman filters uses information from previous frames about the object to formulate a best guess about the current position of the object, and is a popular choice in tracking systems. Several different methods exist today that are highly accurate, given certain restrictions and assumptions.

While the above listed challenges are true for nearly all object tracking systems, the system of interest described in this study is animal tracking; in particular, wild animals in their natural habitats. Several tracking techniques rely on consistency of motion to predict the movement of objects in the presence of full occlusions: the trackers assume that the object continues to move more or less in the same direction and speed as the last few frames and look for the object in the predicted locations. While such methods work well for most classes of objects such as human beings or vehicles, animals often tend to have non-uniform motion – it is not always straightforward to predict the direction of their movement. Moreover, when animals move in groups, it is difficult to distinguish one from the other, and if they cross each other, then tracking them uniquely post the crossover is a challenge if they switch the movement direction. One method to solve such problems is to use identification, or a variant of *re-identification*. Each animal or object is uniquely identified, and even if they cross one another, their visual tracks can be found through the unique identities associated with them.

A background on the project is provided in Section 1.1. A concise problem statement is described in Section 1.2, Section 1.3 briefly describes the steps involved during object tracking, and Section 1.4 introduces the scope and methods used in this study.

1.1 Background

Animal transgression into human settlements is an unfortunate yet common occurrence in India, especially in villages and towns bordering forest regions. Dwindling forest cover and expanding human population cause animals to be displaced from their natural habitats, forcing them to stray close to villages and border areas in search of food. Typically, the transgressions end with intruders attacking and carrying away cattle or feeding on crops. However, there are instances where humans fall prey as well. Almost inevitably, such transgressions of wild animals are discovered only after the first few causalities, and possibly even loss of life. If the forest department action is delayed, the villagers set crude and often ineffective traps, which result in injuries or even death of the animal.

At present, in order to keep cattle and people safe, sentries stay awake through the night to watch for such intrusions and scare away the wild animals. More drastic and dangerous methods are use of electric fences, where the animals get electrocuted on contact. It is a tragic consequence such fences at times causes deaths of people also when they come in contact with the fence by accident. A new alternative is the use of Bio-Fencing [5], where certain types of plants are grown around the borders of farms, which deter animals from entering into the farmland due to their scent.

While such methods can sometimes prove to be effective, the question stands as to how technology can help improve the situation. The approach described in this thesis is the use of a computer vision based monitoring and early-detection system, which can be setup along the borders of the villages to enable triggering of alerts so suitable action can be taken (and hence the project name – *Digital Borders*). This project focuses on building such a system, and the immediate challenge attempted to be solved is animal tracking, where the movement of an animal of interest is tracked through a sequence of images to understand if its movement constitutes an immediate threat to the local populace. Due to use of multiple cameras spread uniformly across the space being, it is necessary to find if animals spotted in different cameras are the same or are different. This is also useful to find if there is a threat from a single animal or multiple. This can be done through identification of animals of the same species, by associating each individual with a unique identifier. A lot of computation required to achieve identification needs to be done locally, potentially on embedded systems such as a Raspberry Pi device, so the algorithms employed need to use low computational power. This is the focus the study described in this thesis.

A concise statement of the problem is provided in the following section.

1.2 Problem Statement

The ideal tracking system needs to work well at all times, and make intelligent decisions to confirm the object's location within the frame. The system needs to draw a tight bounding box around the object of interest (the subject), and in subsequent frames, ensure the bounding box moves in-sync with the subject. It should attempt a maximum overlap of the subject area and the bounding box area, even in the presence of other visually similar objects surrounding the subject. It should work under varied lighting conditions and under change of orientation of the subject. If the subject is partially or completely hidden from camera view, the system should notify the conclusion of tracking, and should not switch targets arbitrarily. If the subject comes back into view shortly, the system must be able to detect this, and continue plotting its trajectory smoothly from the last known point. As can be seen, the specification described here is quite complex to meet, especially for general object trackers, where no prior information is known about the object properties.

While recent object detection methods perform exceedingly well on objects that have consistent or constant motion velocity and direction, animals are more likely to change pose than vehicles, human beings or other inanimate objects. Each different individual within the same species have distinctive features, which help differentiate them (this is not true for some categories of objects, such as two cars of the same make).

The study concentrates on three species of animals, namely **tigers, elephants and jaguars**. The proposed methodology attempts to assign unique identifiers to each individual animal in the set of images or video sequences provided, with the same individual getting the same identifier, even if the images contain the animal in a variety of postures.

1.3 Object Tracking

In Computer Vision, Object tracking is the process of automatically annotating an object with a bounding box, and provided a sequence of images or a video, following the movement of the object of interest in each frame. There is no prior information available about the image(s) regarding the number of objects, or their categories. In addition to drawing bounding boxes over the object, if there are multiple objects detected in the frame, each separate object also needs to be assigned an identifier so that it can be tracked without interfering with the trajectory of a different object.

Tracking consists of several intermediate steps. A short description of each is provided below.

1.3.1 Detection

The first step in an object tracking system, performed on every single frame, is to localize objects of interest in the image, and draw tight bounding boxes around them. This could be a manual input to the first image in the object tracking sequence. However, manual inputs are not possible in a surveillance system.

The objects enclosed in the bounding boxes also need to be categorized into their respective classes. In Deep Learning Networks, detection and classification usually go hand-in-hand, where a region-proposal network is used to generate possible locations of some object within the image, and confidence scores for every known category are generated for each of these regions. A pre-defined threshold is used to filter out regions which have poor confidence. The region acquires the category that has the highest confidence score.

1.3.2 Identity Association

The object enclosed in each bounding box needs to be assigned an identifier, so that it can be distinguished from other objects within the same image. This id is notational only, and has no semantic meaning. Once an id is associated to an object, the same id is reused for each occurrence of the object in each frame of the video sequence. The id and the center of the box can then be used to plot the trajectory of the object as it moves between frames in the video sequence.

1.3.3 Identity Preservation

It might so happen that the object of interest (subject) is not detected in a few frames, either because it has moved behind some foreground object, or it is obscured by the background, or it is partially hidden by a different object. It may re-appear after a short while. The tracker should recognize the subject, and assign it the same id. Similarly, if the subject's appearance changes slightly due to variations in pose or orientation, the tracker should not treat it as a different object.

An *id-switch* is said to occur if two objects of the same class, such as two tigers, overlap in the image for a while, and once they dis-engage, the identities of the two animals are reversed. This may have unintended effects on tracking performance, and a good tracker must continue to assign the correct id to each individual even under such cases.

1.4 Proposed Solution

The methodology for object tracking of wild animals in forest terrain proposed in this thesis works in several stages, and the mechanism used is re-identification of individuals across different images.

Deep Learning is proposed to be used for the object detection, including the localization and classification tasks. Once the bounding boxes for each identified animal has been found, a second Deep Neural Network is used to extract features of the animal in the image, and these extracted features are pre-processed appropriately before being run through a multi-class SVM classifier to associate the bounding box with an identifier.

The pipeline is required to identify three visually and structurally disparate animal species – tigers, jaguars and elephants. Each of these species have different features, and a single system will be used to identify all of them (the mechanism used for each species may be different).

Once the identity has been associated with the individuals, this can be used to track their movement, thereby removing the need for any predictions when it is occluded or moves out of view of the camera. Such a system also prevents id-switch to an extent, since even if the animals cross over since the identification mechanism doesn’t use past movement information to guess the next direction of movement.

Another desired characteristic of the system is the speed of operation – since this needs to be done near real time, the solution attempts to complete the entire identification process in sub-second time interval, and targets achieving speeds of up to 4-5 fps.

2 LITERATURE SURVEY

Use of Convolutional Neural Networks for Computer Vision is fairly new. Once begun, the most important aspect from their use for image processing was automatic feature extraction. Earlier, features were mostly hand-crafted, and as such, rarely transferable across different applications. There were some algorithms and operations that were generic, such as edge, corner and contour detection, but these were only building blocks, and higher level feature extraction mechanisms, through the use of CNNs made image processing more easily accessible to the research community.

For this study as well, CNNs are used for classification of animals, and additionally for extraction of features per individual. The literature survey presented here is split broadly into multiple sections – section 2.1 explores a few of the recent advances in CNNs, followed by section 2.2, which describes a few studies which have focused on animal detection using CNNs. Section 2.3 describes some common techniques used for Animal Re-Identification. A few papers related to Person Re-Identification are surveyed in section 2.4, to underline the similarity of methods used to animal re-identification. Section 2.5 presents some recent works that use Object Tracking.

2.1 Convolutional Neural Networks – Recent Advances

One of the earliest successes that introduced Convolutional Neural Networks for Computer Vision was the work of Krizhevsky et al. in [6], popularly referred to as *AlexNet*. The ImageNet [7] database was used to train the network using nearly 1.2Million images, and classify nearly 1000 different objects. It consists of 5 convolutional and 3 fully connected layers. The network had a top-5 error rate of nearly 17%, the lowest at the time of writing.

Since then, several new architectures were introduced, and CNNs become ever more popular. Girshick et al. [8] introduced a novel network architecture (called R-CNN) that first created region proposals from an image, and then classified each proposal so as to detect the bounding box as well as the category of object in the picture. Subsequently, the authors improved upon the architecture to produce a fast variant in [9]. Fast-RCNN used computation sharing in convolutional layers to speed up the classification. This led to a steep decrease in train and test times. The speed of the architecture was further improved in [10], where attention networks were used and both classification and proposal networks were combined into a single network. This architecture not only further reduced the time taken, but also led to sharp increases in accuracy.

Redmon et al. [11] introduced a, now very popular, neural network called *You Only Look Once*. The distinguishing feature of the architecture proposed here is that the problem of object recognition is treated as one of regression, rather than pure classification. The architecture just requires only one pass of the image to both generate the bounding-boxes for objects, as well as classify them (hence the name). The advantage this approach carries is significant -- a single network does both tasks, therefore largely reducing test and training times. *YOLO* also has additional modes of operation, such as *Tiny-YOLO*, which uses a much smaller neural network. While the basic network works at a frame rate of 45fps, the smaller network produces nearly 150fps (on poweful GPUs). The solution achieved a 58% mAP score on the PASCAL VOC 2012 dataset, which was in the top 15 state-of-the-art systems at the time.

Redmon et al. continue their work on *YOLO*, extending it in [12] and [13] for incremental accuracy gains, while retaining most of the previous performance. [12] is capable of classifying 9000 different categories, up from the 20 categories recognized by [11]. It was able to obtain an mAP of 78.6% on the PASCAL VOC 2007 dataset. [13] is another improvement over the second version, which increased the number of layers in the network to 53 from the existing 19, and had several other design changes, was able to achieve near state-of-the-art accuracy, but at much faster FPS.

In an effort similar to that of YOLO, Liu et al. present Single Shot Detector [14], which uses a single neural network and a single pass to detect both the object category as well as the bounding boxes. The notion used here is to vary the size of the feature maps, as opposed to varying size of the region proposals. The resultant network performed faster than YOLO or Fater R-CNN, while obtaining a better accuracy.

2.2 Animal Detection and Classification using CNN

The article by Norouzzadeha et al. [15] uses Deep Neural Networks to detect and count wild animals in camera trap images. Additionally, the network they propose also identifies some attributes of the animal, such as sitting or standing, eating etc.. They use the Snapshot Serengeti dataset [16] for their study, which was the largest collection of annotated wild animal images at the time of writing. Since a large portion of the images in the dataset did not contain animals of images (false triggers), their solution used a 2-stage pipeline - one neural network detected the presence of an animal, and a second network classified the species and characteristics, if an animal was found. Their solution was also focused on classifying single species per image, so any images which had multiple species of animals were rejected. Various architectures were tested, including AlexNet [6], VCGNet [17], GoogLeNet [18] and ResNet [19]. For three tasks - animal detection, species classification and counting, they found that different networks had the best score. For this reason, an ensemble of models was trained, and it performed better than any individual network in all three tasks, achieving an species classification accuracy of 99.1% (top-5), and a counting accuracy of 84.7% (+- 1). While this approach is highly accurate, it is also highly resource intensive.

Verma et al. [20] in their paper, study classification of animals from camera trap images as well, but under specific cases where there is a clutter in the picture caused either by animal movement or number of animals. Their solution works by first generating region proposals from the input image which are likely to contain animals, and the using these proposals to classify the animals. The network they use is similar to that in [6], The feature vector obtained from the CNN is then classified using K-Nearest-Neighbors for an average accuracy of around 91.4%.

Feng et al. [21] use a classic approach for animal monitoring, with the use of Saliency object detection and contrast histograms. This study uses images acquired from a wireless multimedia-sensor network instead of camera traps, with image sensors forming a mesh type network to capture and transmit images to the edge server. The images are therefore acquired from a wide variety of terrain, as opposed to the generally fixed backgrounds of camera trap images. The precision rate of their work was close to 49%.

Matuska et al. in [22] build an animal monitoring system to help detect animal migration corridors. This approach doesn't use neural networks, and is based on extracting scale and translation invariant features from detected objects and classifying them. The key point extractions of the object are done through SIFT [23] and SURF [24]. For classification, a combination of Bag-of-Visual-Words and Support Vector Machines [25] were used. The training dataset consisted of 5 animal species, and a total size of 300 images. The best combination of key-point extraction and classification yielded an accuracy of 94%.

Zhang et al. in [26] address the problems related to mistaken identification of background as animals in camera trap images. To overcome this, each animal region proposals are generated using multilevel graph cuts, and are verified through cross-frame temporal matching. Only those regions that pass the verification are then used for training the DCNN. For classification, the accuracy is attempted to be improved by combining auto-extracted object features as well as using Histogram of Oriented Gradients (HOG) features and Fischer Vectors. An average precision rate of around 83% is achieved over the custom camera-trap dataset used.

Nguyen et al. in their [27], propose a framework to help easy animal recognition in the wild, when using camera-trap images using the Wildlife Spotter dataset. Their system consists of a 2-network pipeline -- the first network detects the presence of an animal, and the second network identifies the species. The best detection accuracy from the employed CNNs was close to 97%. Classification accuracy on the dataset was close to 88%.

Wilber et al. [28] present their work on building vision tools for field biologists to study the endangered desert Tortoise and Mojave Ground Squirrel. The algorithm they present is unique in the sense that it is suitable to run on low powered device such as a smart phone. The article uses SIFT [23] for feature detection, and uses an SVM classifier for species identification. They achieved an accuracy of around 78%.

2.3 Animal Re-Identification

There are several approaches that could be used for animal re-identification. In this survey, they are broadly grouped into categories that use classical computer vision techniques - such as SIFT, manual feature extraction, Histogram of Oriented Gradients etc., and those approaches that use CNNs.

A survey paper is presented by Schneider et al. in [29], which describes nearly 30 papers related to animal re-identification. Most papers presented below are included in the paper, but there are a few additional works covered here as well.

2.3.1 Classic Computer Vision Techniques

The articles listed in this section are further split into multiple categories - some approaches use hand-crafted feature extraction, some use SIFT to extract features, and yet more use contrast maps or texture maps for feature extraction.

2.3.1.1 Using histograms, contrast maps

Perez et al. [30] in their article look at how small animals in lab environments are tracked between frames. One of the challenges when dealing with several small animals in contained environments is that they constantly cross paths with each other, creating partial or full occlusions. It is important to re-identify the correct animals once such occlusions occur. The method described here does so using combinations of contrast maps and intensity maps to fingerprint each individual animal. Once fingerprinted, the animal can be re-identified by the system, even if it has completely disappeared from the FOV of the camera, and re-appears at a later point. The accuracy of this approach was very high - close to 99.6%. Given that the setup was a lab environment, advantages of fixed backgrounds and proper lighting helped remove most problems generally associated with field photos.

Rodriguez et al. [31] present *ToxTrac*, a software system that helps image based tracking of organisms monitored in lab-based environments. The solution proposed is fast, working at 25FPS even with a full HD resolution input. The animals are segmented using background subtraction. Once the animals in the image are segmented, the characteristic features of each individual are calculated using intensity histograms, and texture center maps. Body tracking is performed using a Kalman Filter. The method was tested in the lab for eight different species, such as fish, ant and mice.

In order to monitor penguins non-invasively, Sherley et al. [32] describe a computer vision system that uses the chest plumage to uniquely identify a penguin. The proposed system uses a distortion specific distribution context (Shape Contexts) to extract features from each image, and compare them with the known image features. The system achieved an acceptance rate of nearly 97%.

2.3.1.2 Using SIFT for feature vector generation

Crall et al. in their article [33] describe a system that helps identify animals using their coat patterns, such as Zebra, jaguars, giraffe etc. The system is partially automated, in that the bounding boxes for the coats of the animals are extracted manually. The feature descriptors are calculated from the ROIs using a variant of the SIFT [23] algorithm. Matching of labels is done through one-vs.-one or one-vs.-all matching systems, to find distance between stored images in the database and calculate most similar ones using Local Naive Baye's Nearest Neighbor approach. Failures in matching were primarily due to variations in pose and image noise.

In order to aid jaguar protection from poaching, Dhulekar et al. [34] propose a system of automatically identifying a jaguar given its image, that works even for noisy images or night-time images. The jaguar is automatically segmented from the image using a static threshold, and a Histogram of Oriented Gradients is used to check if the image contains a jaguar. Once the jaguar is segmented, SIFT descriptors are extracted for the jaguar, and compared with known jaguars to check if any match. Over a dataset of around 150 images, they achieve an accuracy of 89%.

Town et al. present *Manta Matcher* [35], which is a software to uniquely identify a manta ray given an image. Once a picture of a manta ray is taken, the bounding box of the image is then manually indicated, after which the SIFT features for the bounding box are extracted and converted to vectors. These vectors are compared with the stored feature vectors, and a probability of match is returned. This approach resulted in a top-10 accuracy of nearly 90%.

2.3.1.3 Using hand-crafted features

Hiby et al. in [36] describe their work on building a software to automatically identify a tiger from its stripes. The curved flanks of the tiger are flattened first, and multiple features are extracted (such as legs, tail, hips etc.), and a mapping is created out of these points. Two algorithms are then used to match tigers, one of which is robust to noise, and the second robust to occlusions. A combination of these two algorithms helps matching the images of the tigers.

Kelly in [37] describes a system that helps ecologists identify a jaguar in the captured image. A computer aided 3-D matching system is demonstrated to identify individuals. The flank or ROI of the animal is manually specified, and the correlation co-efficient is used to measure similarity among two jaguars. Minor variations in pose or lighting could be handled by translating the feature vectors in different directions. This approach achieved a high accuracy in identifying individuals.

Lahiri et al. [38] in their work describe a method named *StripeCodes* to identify individual zebra using their stripe patterns. The coat patterns are manually extracted from the query image, and are converted to a *code*. These codes are then compared using edit distance to find individuals which have similar stripe codes.

The article by Ardovini et al [39] describes a method to identify elephants using multi-curve matching. The ears of the elephants, the shapes of the curves, and the nicks in the ears are used to identify the elephant. 2-3 reference points are selected by the user from the query image, and the features of the ear curves extracted from these references are matched with those present in the DB. This system was able to achieve 50% top-7 accuracy.

2.3.2 Re-Identification using CNN generated feature vectors

Li et al. [40] discuss re-identification of Tigers whilst releasing a new annotated dataset to the public. The dataset has been manually annotated using stripe patterns on the tigers. The paper describes a novel method to identify tigers, through pose estimation. Several key-points of the tiger's frame, such as hip, legs and tail are used to estimate the pose, which is used as a distinguishing feature among the tigers. The proposed solution uses a ResNet50 network as a backbone to detect bounding boxes of the tigers, and then use a triplet-loss metric for feature identification. They achieve a 74% accuracy with re-identification when detecting tigers in the plains. However, in wild regions, their method does face difficulties extracting features for pose estimation.

The focus of the paper [41] is on identification of elephants. The article uses YOLO [11] to first detect the bounding boxes around the elephants, and the bounding boxes are run through a ResNet50 [19] network to extract features. The features are then classified using an SVM [25] classifier to assign the most probable label. The bounding boxes are placed over the heads of the elephants only. Over a dataset of around 2000 images, they achieve a top-1 accuracy of 49%.

The study by Deb et al. [42] focuses on primate face recognition to assist in counting and monitoring endangered primates, and it used images of lemur, golden monkeys and chimpanzees as subjects, which are then used for species and individual identification. The images were manually annotated to localize landmarks, such as fur, eye-shape etc. The neural network introduced here, called *PrimNet* is based from the SphereNet CNN.

Cheema et al. in [43] describe a method that attempts to identify multiple species with distinct patterns on their skin. The study included tigers, zebra and jaguars. They use a Faster-RCNN [9] neural network to extract bounding box proposals for the animals in the image, and then run the bounding boxes through an AlexNet [6] network to extract distinct features. A PCA is done on the feature vector, and label is associated to the animal using an SVM classifier on the features. They achieve good accuracy on Tiger and Zebra (80% and 93% respectively), but obtain a lower accuracy of 78% on jaguars. The result indicates that while the system performs well on animals having striped coat patterns, it under performs for animals having spotted coat patterns.

2.3.3 Re-Identification using Face Recognition

Bergamini et al. in [44] describe a system for cattle re-identification. Two different CNNs are trained with multiple views of faces of the same cattle, and the outputs of each network are combined to form a single feature vector. A K-Nearest-Neighbors classifier is then applied to find the label of the animal. They obtain nearly 82% top-1 accuracy in closed set identification.

Brust et al. [45] study the transfer-ability of human facial recognition algorithms to identify Gorilla. The dataset they use consists of gorilla photographed using camera traps, use the YOLO network to find bounding boxes (over the faces) of the gorilla in the image, and subsequently run the extracted bounding boxes through an AlexNet network for feature extraction. SVM classifier is used for associating labels to the images. The study achieves top-5 accuracy of around 80%.

Freytag et al. in [46] describe a method to automatically identify multiple attributes for each chimpanzee, such as a unique identity, age, gender etc. The system introduces use of a Log-Euclidean framework on top of bilinear pooling in CNNs to extract fine variances in features of the chimpanzees. An AlexNet network was used to extract the feature vector that identifies the chimpanzee. On a high-quality dataset, the system achieved an accuracy of nearly 92%, but the accuracy drops to around 75% when the dataset included images of poorer quality.

2.4 Person Re-Identification

Person re-identification is not a new subject. It has been extensively explored over the last 2 decades, and several advances have been made in the field, especially in the context of surveillance. Person re-identification bears a close resemblance to animal re-identification, and a few papers that describe human re-identification from images are described in this section.

Zhang et al. introduce *AlignedReid* [47], which is a novel method that uses both local as well as global features during training for human re-id. In the training phase, their neural network jointly learns both local as well as global features of each image to identify the person. The persons in the image are split along horizontal bars, and then matched with the existing images in the dataset. The shortest distance between these horizontal bars is then used to establish a match between the pair of images being compared. A weighted similarity is assigned so that corresponding parts of the image contain more weight, than the non-corresponding parts. The method described achieves an astounding top-1 accuracy of 94.4% on Market1501 and an accuracy of nearly 98% on CUHK03, two publicly available person re-id datasets.

Zheng et al. describe a method for person re-identification dataset, as well as a novel algorithm for person detection and recognition [48]. The system first detects the presence of a person using a CNN, and if detected, the person's image is passed through an AlexNet for feature extraction. The extracted feature is then matched using a confidence weighted similarity scoring system. The paper also recommends use of an IOU of 0.7 over the more typically used 0.5 for better results in person re-identification.

2.5 Object Detection and Tracking

There exist two popular contests organized to encourage and benchmark object tracking systems – namely *Visual Object Tracking* , and *Multi Object Tracking*. Both challenges provide the dataset to work with, which has several varying scenarios and contain detailed evaluation metrics. The results are published on a yearly basis, and the list of papers compiled below mostly refers to one of these two challenges.

A good survey of current multi-object tracking systems has been presented in [49]. The paper also provides a detailed explanation of the metrics commonly used to evaluate multi-object tracking systems. The paper lists Faster-RCNNs and Single Shot Detectors (SSD) as some common object detection networks. The paper also goes on to describe that CNNs are now widely used for feature extraction of objects over other conventional methods. Recurrent networks such as Siamese Neural Networks and LSTM are used to learn to differentiate between features extracted from two objects of the same class. The survey also highlights that the major difficulty today in object tracking setup is to achieve real-time and near-real-time performances with high accuracy. It also shows examples of Id-Switch, which is faced even by the best performing trackers. The survey concludes stating its observations that Deep Learning and CNNs are essential for feature extraction, and highlights the importance of accuracy in the object detection step to improve performance of the subsequent parts.

The results of the seventh Visual Object Tracking Challenge, VOT 2019 are presented in [50]. The challenge considered the 81 trackers submitted for evaluation as part of the challenge, and posts the results of the evaluation over multiple different metrics. The state-of-the-art in short-term tracking was the DRNet system, which uses a two stage framework – one to estimate the target position, and a regression network to localize the bounding box positions. *ResNet50* is the backbone network used, and a Discriminative Correlation Filter is applied to improve the accuracy of prediction. Other techniques in the top-3 include the use of triplet-loss and SiamMask for object segregation.

The current state-of-the-art in the MOT2019 challenge is the ODESA system, from the Samsung Research and Development Institute, Ukraine [51]. This system uses a ResNet101 backbone network, and uses Faster-RCNN for object detection. The Hungarian Algorithm was used in combination with Kalman Filters for object association.

Wang et al. describe a system called SiamMask in [52]. The system is initialized with a single bounding box at the beginning of the tracking sequence, and produces segmentation masks as the output of each image frame. It is extremely fast, and performs at 55fps. The proposed method uses a fully convolutional Siamese Network that reads in pairs of images during training, and uses a custom loss function to optimize training error. A ResNet50 base architecture is used for each CNN. The method achieves an mAP of 0.9 at 0.5 IOU, and 0.6 at 0.7IOU.

Another method that uses Siamese CNNs is [53], where they use the concept of rotated bounding boxes to handle deformable objects in tracking. Their method is the successor of the SiamMask method to estimate the bounding boxes, and in addition, the rotation angle is also estimated to better find the bounding box position and the maximum ground-truth overlap. They achieve a good accuracy of 0.65, which sets the State-of-the-art for the VOT2018 challenge.

A recently popular approach to multiple-object-tracking is the Deep SORT method [54]. The Deep SORT is an improvement over the Simple Online and real-time Tracking (SORT) algorithm, which is a detection and tracking framework that achieved state-of-the-art performance on the MOT dataset. They implement a cascade that solves a series of measurement-to-track associations to optimize the tracking of individual objects in a local context. To extract feature descriptors, they use a CNN that has been trained over a person re-id dataset and uses a residual network. This has directly constituted to a nearly 45% decrease in the number of identity switches, while maintaining a frame rate of 20Hz on a GPU.

2.6 Summary

This chapter briefly describes the most relevant works to the problem being addressed in the project. Since there are two parts to the solution -- one to detect and classify the animal in the image, and the second to identify it, the survey presented here delves into related works for both. In addition, some recent works in the field of Object Tracking are also presented here, along with some background information regarding CNNs. A summary of the pros and cons of each approach for animal re-identification are tabulated in Table 2-1.

Object tracking has made significant progress in the last few years, and nearly every competent object tracking system uses a CNN for object detection as well as feature extraction. Some even use multiple CNNs for the feature extraction phase, to train the networks to correctly extract distinctive features even for intra-class objects.

Approach	Pros	Cons
Hand-Crafted feature extraction	• Not resource-intensive	• Generally require some manual input • Species-specific, hence difficult to generalize
Using SIFT etc. for feature extraction	• Features are highly scale and illumination invariant • Not resource-intensive	• Generally require some manual input
Using Histograms, Contrast / Texture maps	• Not resource-intensive • Works extremely well in controlled environments	• Affected by changes in illumination • Very difficult to differentiate macroscopic features
Using Facial Recognition	• Extensively explored domain, so several challenges already addressed	• Typically need to train network to recognize faces - not possible for wild animals • Would require high resolution image captures
CNNs for feature extraction	• Features are auto-extracted, and are highly discriminative • Species Classification is simple and accurate	• Would mostly require two neural networks - one to classify, one to identify • Features extracted from one layer may not work to identify all species

Table 2-1 Summary of different design approaches for Animal Re-Identification, with their pros and cons.

3 System Requirements Specification

This chapter describes the system to be designed for tracking and identification of wild animals as a whole, and provides a bird's eye view of the proposed solution.

The main intention behind developing the system is an attempt to solve the problem of the human-animal conflict, which especially arises in villages and towns bordering forest areas. This study proposes to establish a network of cameras, which are placed at strategic locations around the borders of the settlement to continuously monitor the surroundings for presence of wild and dangerous animals. If such animals are detected, they are tracked for a suitably long time, and once the system ascertains the intent of the subject, create and publish alerts to an appropriate authority or a local sentry for manual intervention.

3.1 System Visualization

The futuristic visualization of the system is illustrated in Figure 3-1.

3.1.1 Cameras

The diagram shows a system of four cameras, each connected to a processing unit. The animal would only be detected if it moves into the Field of Vision (FoV) of the camera device. The FoVs of the cameras may or may not overlap, depending on the requirements specific to each site. The dotted line in the picture on the left indicates a possible movement trajectory for the animal.

3.1.2 Processing unit

The processing unit is a compute device, such as a Raspberry Pi or a similar microcontroller. The function of the processing unit is to read images from the camera, and process the images to check for the presence of a wild animal of interest, such as a tiger or an elephant. Each processing unit operates completely independently, and is expected to have a self-sufficient power source. Each processing unit contains a communication device, preferably wireless, that can relay information to a remove server.

3.1.3 Centralized Server

The server monitors the entire system operations and is connected to a database, which can store important messages and data related to each site's operation. The server is also responsible for tracking the animal across multiple cameras, since once the animal moves out of the FoV of a camera, the processing units couldn't continue tracking the animal.

3.2 Functional and Non-Functional requirements

The system described in Figure 3-1 is the tentative visualization of the end goal that this thesis aims at. It is essentially the application or use case that the methods proposed here target. The immediate goals of the software required to facilitate the solution are described in this section.

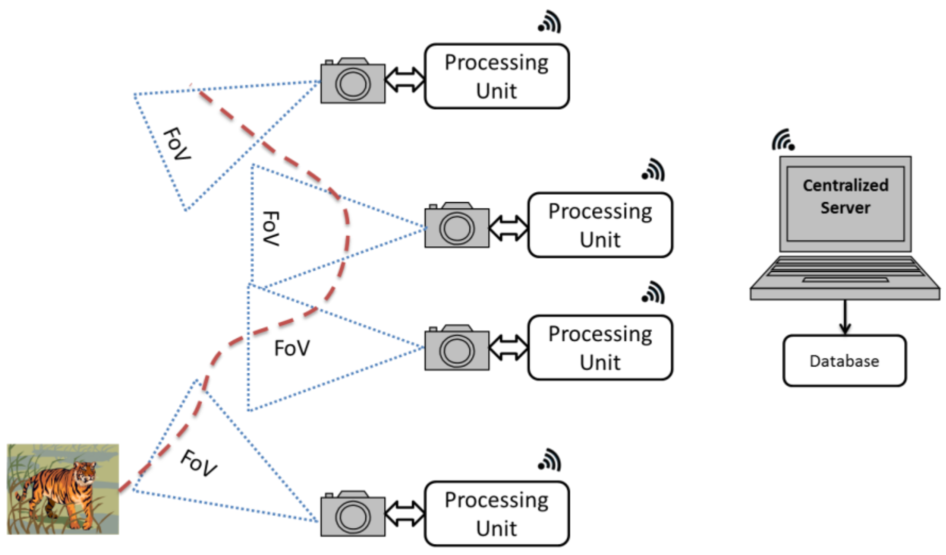

Figure 3-1 Futuristic view of the proposed solution; the red line indicates the trajectory of movement of the animal. FoV stands for Field of Vision. Each processing unit is capable of communicating with a remote computer, which is responsible for further processing and monitoring.

3.2.1 Functional requirements

F1. Given an image, the system should be able to detect if it contains one or more tigers, elephants or jaguars.

F2. If the given image contains an animal of interest, the system should localize the detection to a tight bounding box around the animal, and classify the animal contained in the bounding box to the correct species.

F3. Once an animal is detected, the system should register the animal and assign it a temporary unique identifier.

F4. If the system is presented another image of the registered individual animal, possibly captured a short while after the registration, the same individual needs to be assigned the same identifier.

F5. Using the identities assigned to the animal across a sequence of video frames, it must be possible for a tracker to be able to accurately follow the movement of the animal.

F6. The duration of retention of the unique identifier for an individual could be in the order of minutes. If the same individual is detected after a gap of a few days post the completion of tracking, the identifier assigned to it need not be the same as before.

3.2.2 Nonfunctional requirements

N1. The system needs to be accurate in the classification of the species of animal.

N2. The bounding box localization generated by the system to locate the animal within the image frame must be as close to the ground truth bounding box as possible. The common metric used to measure this, Intersection-Over-Union, needs to be at least 0.5

N3. The identification mechanism used by the system must be robust. The same individual could re-appear in different poses, angles, or under different light illuminations. The identification must succeed in such varying visual conditions.

N4. The system needs to be fast. Since potential applications of the identification mechanism could be in real time, the entire identification operation must run at speeds of at least 3-5 frames per second.

N5. The system needs to be fault tolerant. Any failures during the localization phase or identification phase needs to be handled gracefully with meaningful errors being displayed to the user.

3.2.3 Constraints and Assumptions

There are several difficulties that arise in complex systems due to a multitude of factors. The constraints set in place for operation of the solution described in this study are listed below.

a) The study targets three different species of animals – tigers, jaguars and elephants. Other species would neither be detected nor tracked by the system
b) Only daylight images are considered for the initial phase. The camera is a visible light sensor, and the resolution of the camera is good enough to capture images under a wide range of lighting conditions.
c) The camera is positioned such that there is a clear view of the surroundings that it can capture. There should not be bushes or trees obstructing its view
d) If the animals move only along the edge of the FoV of the camera, the resolution of the camera may not be good enough for the system to detect it.
e) The system works if there are a small number of animals in the image, ideally only a single animal of a given species. Groups of animals are not being tracked, but one of the individuals in the group would be tracked
f) The system is being targeted to do a large portion of the image processing on embedded and low-power devices. Appropriate algorithms are used and procedures that are highly computationally intensive are avoided.
g) Since the communication is wireless, any communication between the devices or between the device and the server are performed through light-weight protocols such as MQTT.

3.3 Test Case Specification

The test cases run for various parts of the code are listed below.

3.3.1 Object Detection Tests

Test Case Number	Test Case Name	Description	Test Case Type
T1.	Blank Image Test	Given an image containing no foreground objects, system returns 0 bounding boxes	Negative
T2.	Single Animal	Given an image containing exactly one animal – a tiger, jaguar or an elephant, the system returns one bounding box, at the same location of the animal in the image, with IOU >= 0.5	Positive
T3.	Multiple Animals, Same Species	If the animal contains more than one animal of the same species, the system returns multiple bounding boxes, each covering erringly one animal, max 10 animals per image	Positive

T4.	Multiple Animal, different species	If the image contains animals from multiple species the system detects and classifies each of them correctly, and placing bounding boxes appropriately	Positive
T5.	Poor Lighting	The system should be able to detect presence of the animal under poor lighting conditions, and accurately place the bounding box over the animal	Negative
T6.	Partial Occlusions	If the animal is partially occluded in the image (< 25%) then the system should be able to detect the animal and place bounding box at the correct location	Positive
T7.	Total Occlusion	If the animal is occluded by more than 50%, the system need not detect it	Negative
T8.	Performance Test	The object detection completes detection in under 150ms on average	Performance

Table 3-1 Object Detection Test Cases

3.3.2 Object Identification

Test Case Number	**Test Case Name**	**Description**	**Test Case Type**
T9.	Closed Set - Similar poses	Two images of the same individual should be assigned the same identifier by the system, if they are in similar poses	Positive
T10.	Closed set – Differing poses	Two images of the same individual must be assigned the same identifier, even if they are in different poses or orientations	Positive
T11.	Closed Set – Different individuals	Two images of different individuals of the same species must be assigned different identifiers, even if they appear visually similar	Negative
T12.	Open Set	If the individual in the image is not known, an identifier must be assigned to it which has not been previously reported	Negative
T13.	Duration Test	If the same individual is presented to the system after a duration of 2-3 days, it should be assigned a different identifier	Negative
T14.	Performance Test	The identification process must complete in under 150ms on average	Performance

Table 3-2 Object Identification Test Cases

3.4 Requirements Traceability Matrix

Requirement Identifiers	**Reqs Tested**	**F1**	**F2**	**F3**	**F4**	**F5**	**F6**
No. of Test Cases	14	5	2	1	2	7	1
T1		X				X	
T2		X				X	
T3		X				X	
T4		X				X	
T5		X	X			X	
T6			X				
T7						X	
T8							
T9				X			
T10					X		
T11					X		
T12							
T13						X	X
T14							

Table 3-3 The Requirements Traceability matrix mapping test cases to the functional requirements

3.5 System Environment

For the initial phase, the entire system will be run on a laptop computer. The solution will be moved to an embedded device in the next stage of the project.

3.5.1 Hardware Requirements

The minimum requirements to run the software developed as part of this study is summarized in Table 3-4.

Processor	Intel Core-i3, 2GHz
No. of CPUs	4 (2 physical CPUs, with hyper-threading enabled)
RAM	8GB
Cache	L1 Cache: 32KB, L2 Cache: 256KB, L3 Cache: 3MB

Table 3-4 Hardware requirements for the proposed system

3.5.2 Software Requirements

The platform and software frameworks used are tabulated in Table 3-5.

Operating System	Ubuntu 18.04, 64bit
Development Language	Python 3.6
Deep Learning Framework	Tensorflow (Keras Backend)
Image Processing utilities	OpenCV
Third-party software and libraries	Numpy SciKit Matplotlib
Database	MongoDB

Table 3-5 Software requirements for the proposed system

4 PROPOSED METHODOLOGY

The study presented in this thesis describes a novel method of animal re-identification, which is intended to be used for tracking. Apart from this, the system also targets three different animal species to be tracked, each species having a different set of distinguishing characteristics. For example, tigers are recognized by their stripes, jaguars by their spot patterns, and elephants have distinct shapes of ears, among other unique features. To the best of my knowledge, this is the first such system that targets identification of a diverse set of species in a single system. As described in the Chapter 2, there are studies in the past that target one or more species individually, but no system has tried to include diverse species as part of its goal. The closest such system is described in [43], where Zebra, Tigers and Jaguars are identified by coat patterns.

The architecture of the system is described in 4.1. A more detailed view of the system is presented in 4.2, followed by a detailed description of the software design in 4.3.

4.1 System Architecture

The overall system architecture is illustrated in Figure 4-1. The architecture shows an example of the system setup with two cameras, each connected to a micro-controller. One example of a microcontroller is the Raspberry Pi [55]. The micro-controller communicates through one or more wireless interfaces with an optional Edge Server. The edge-server acts a buffer between the centralized server and the device network, thereby reducing latency of communication and also reducing load on the central server. In cases where there is no internet connectivity, the functions of the central server can be completed by the edge server itself.

The centralized server contains a database necessary to carry out all monitoring related operations, and also receives information about the current state of each device. This can be used to alert users if one of the devices malfunction or are running low on power. Tracking animals across multiple cameras has to be done either by the central server or the edge server, by aggregating detections from each camera.

The communication protocols between the devices and the edge or central server would be some lightweight protocols such as MQTT or CoAP. There could be short bursts of transfer, since the devices would be idle a majority of the time.

4.2 System Design

The design of the system presented in Figure 4-2 contains details of each individual block displayed in the architecture and describes their function.

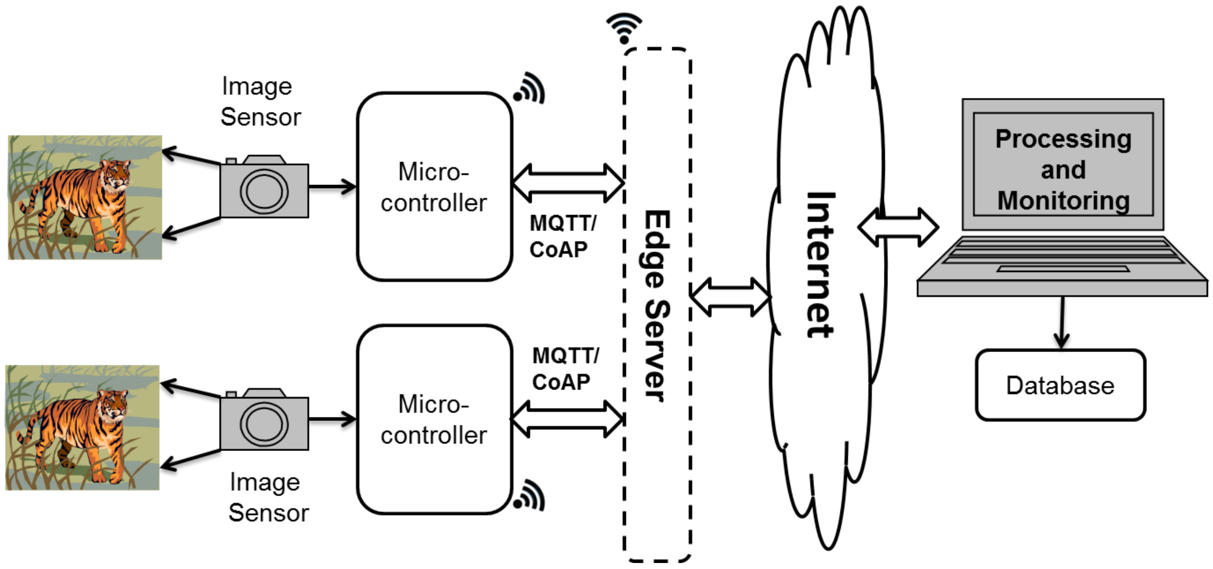

Figure 4-1 Overall system architecture with an example containing two image sensors (cameras) each connected to a micro-controller.

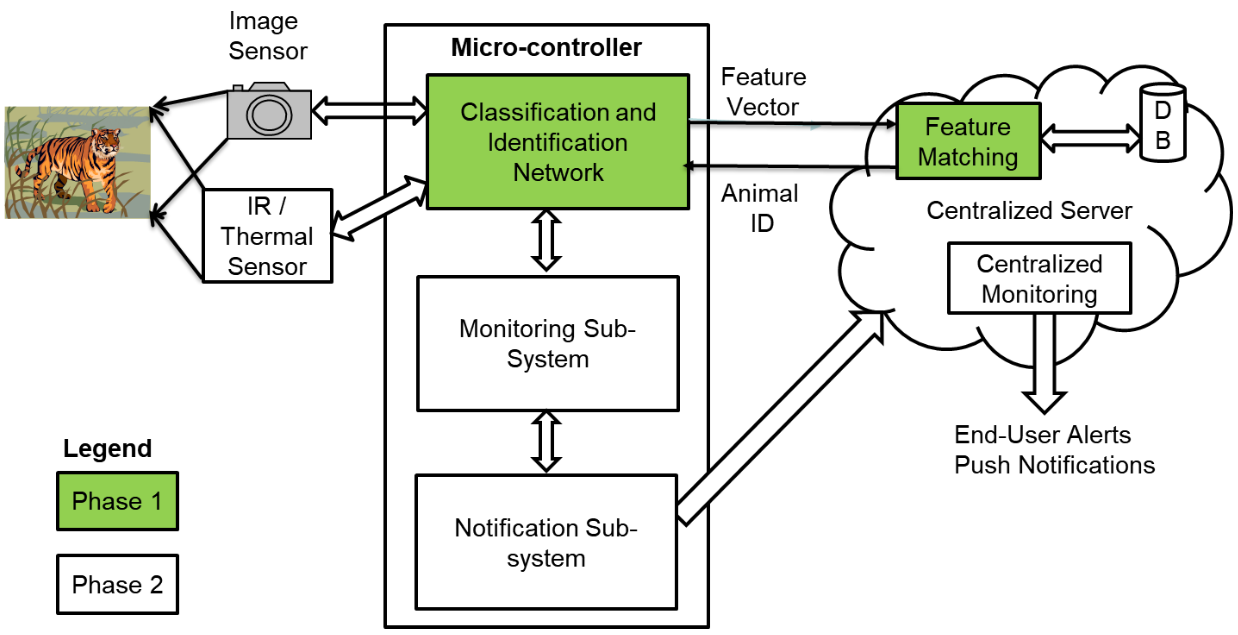

Figure 4-2 Design of the proposed system. The function of each block is illustrated, and the different phases of the project are also shown in different colored blocks

4.2.1 Image Sensors

The function of the image sensor is trivial – its job is to capture images in a good resolution and the images captured need to be sharp, rather than blurry or noisy. Raspberry Pi compatible camera modules, called PiCamera [56], are readily available in retail stores. PiCamera integrates into the Raspberry Pi out of the box, and can capture images up to 5MP in resolution. In the future, IR image sensors can also be used along with visible light image sensors to help the system detect animals during the night or in poor light conditions. The algorithms for this camera might be different compared to the ones described in this thesis.

In order to save power, the image sensors can be integrated with a motion sensor, set to trigger only when motion is detected. This prevents the need for the system to continuously monitor every image captured by the cameras.

4.2.2 Micro-Controller

The microcontroller is the primary compute unit in this architecture. It carries out several important functions, and needs to have sufficient power to run object detection programs. Each image captured by the sensors is first analyzed by the microcontroller to detect if there are any animals of interest in the image. If an animal is detected and its class recognized as one of the three species targeted in this study, the bounding boxes for the object are found, and it registers the animal in its memory.

In order to associate an identity for the animal in the frame, the microcontroller communicates with the central or edge server. The features for the animal(s) in the bounding box are extracted and sent over the network to the server, which assigns an identifier to it. The monitoring sub-system then uses this identity to track the animal and find the direction of movement.

If the animal moves out of view of the camera attached, then the controller sends a notification via the Notification sub-system to the server with the information about the last timestamp of the animal detection, and its general direction of movement. This essentially signals a hand-off, where the controller dispenses responsibility of the subject in question, and resets its internal state and proceeds to wait for the next animal to track.

4.2.3 Centralized Server

The centralized server could potentially be hosted on a cloud provider like AWS, or it could be a site-local server. The server requires fair amount of computing power, and perhaps even a GPU to perform neural network operations. The main responsibility of the server is in two parts – assigning identities to each animal detected, and tracking the movement of subjects across different cameras.

The flowchart of the operations performed by the server when it receives a feature vector from a controller is shown in Figure 4-3. Each time a micro-controller on site reports the beginning of tracking an animal, the server stores the time of sight and the feature vector into its database, and returns an identifier to the controller. This identifier could be that of a previously encountered animal, or a completely new identifier. If it is found that the features received closely match the features stored for some animal of the same class in the database, it reuses the identifier stored in its database. If no close match is found, a new identifier is generated, and stored in the database along with the feature vector. These stored identifiers and feature vectors are deleted after some pre-defined interval, which is in the order of a few hours.

It is likely that the animal that is being tracked moves across the FoVs of multiple cameras placed in the system. So if the server finds a matching feature vector for a subject that is requested from a controller, and the timestamp of both sightings are separated by small intervals, then it is likely that the controller is looking at the same individual that has been spotted recently. This information can be used to track the movement of the subject across cameras.

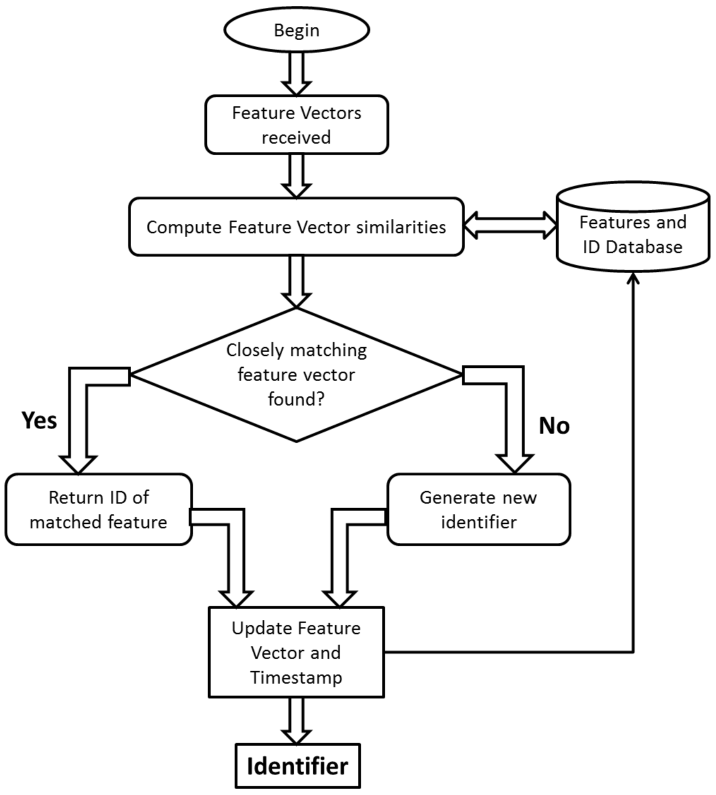
Begin
Feature Vectors received
Compute Feature Vector similarities
Features and ID Database
Closely matching feature vector found?
Yes
No
Return ID of matched feature
Generate new identifier
Update Feature Vector and Timestamp
Identifier

Figure 4-3 Flowchart for the id association operation in the server

4.3 Software Design

The overall approach to the solution consist of four steps –

1. Object Detection and Classification
2. Feature Extraction and Matching
3. Identity Association
4. Tracking movement.

Of these, tracking is planned to be completed in the next phase of the project. The three steps and various design alternatives for each of them are summarized in Figure 4-4.

The overall design and the methodologies used in this study borrow several concepts from the papers surveyed and presented in chapter 2. Most notably, the pipeline and ideas presented here bear strong resemblance to two papers - [43] where the coat patterns of animals are used to identify individuals, and the paper [41], where elephants are identified using a procedure very similar to the one described here. The paper [46] also uses a system similar to ours to run facial recognition on Chimpanzees. The methodology used in this paper combines concepts from all of these, and a few new ideas are included in the mix to yield the new procedure described below.

4.3.1 Object Detection and Classification

Object Detection is the term used to refer to the process of finding portions of the image which contain recognizable entities, which could be one or several. Essentially object detection consists of two parts: finding portions of the image that are of interest, and then classifying these locations to one of the previously known categories. An example of the result of running object detection over an image is illustrated in Figure 4-5.

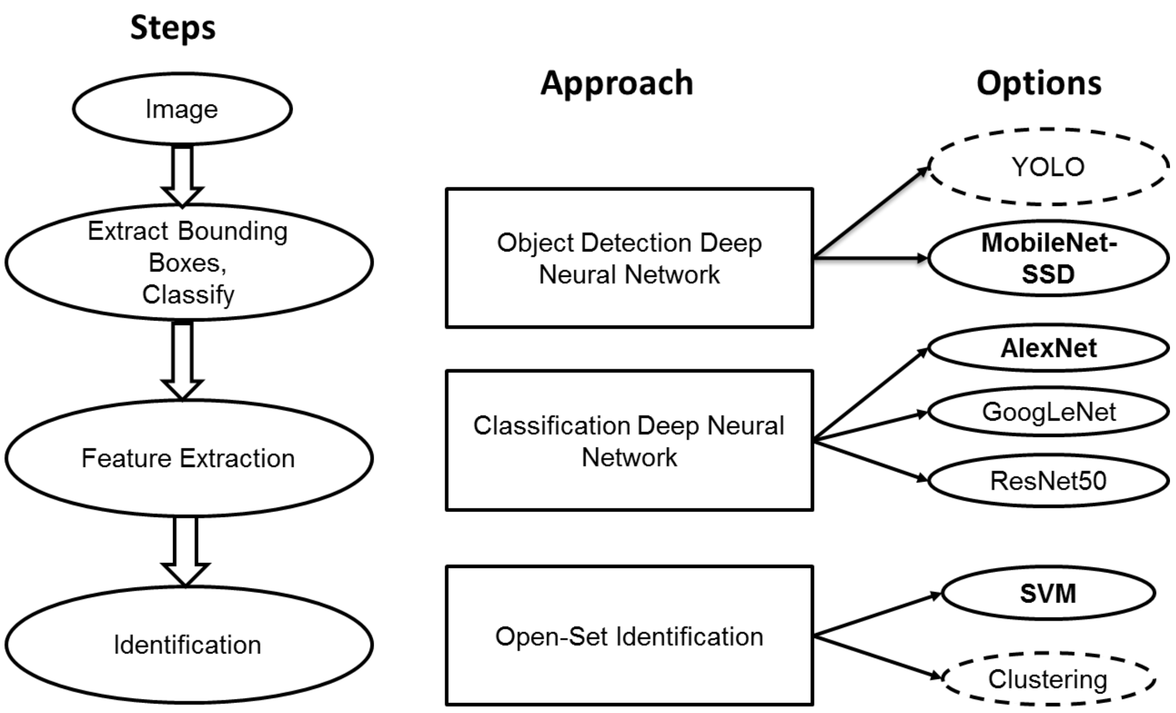

Figure 4-4 Design alternatives for the steps in the tracking process. The approach used for each step is shown to its immediate right, and the alternatives for each approach are shown in ellipses. Dotted outlines indicate that he option is not currently under consideration

Object detection can be performed in a number of ways. In recent years, the clear method of choice is the use of Deep Convolutional Neural Networks (DCNN) for object detection and classification. DCNNs outperform all conventional methods by a significantly large margin in terms of accuracy. A summary of recent advances in deep learning with respect to Object detection was covered in the Literature Survey section. YOLO, Faster-RCNN and SSD are some of the most commonly used object detection systems currently in use today. In this project, SSD is chosen as the object detection model. A brief summary of the Single Shot Detection (SSD) model is provided below.

4.3.1.1 Single Shot Detection

The Single Shot Multibox Detector (SSD) was introduced in 2016 by authors Liu et al. SSD uses a single deep neural network to perform end-to-end object detection and classification given an input image. The core idea behind the SSD detector is the use of feature maps of different scales to produce regions of interest in various aspect-ratios, and each of these is checked for the presence of a known class. The predictions are also separated by aspect ratio, so that overlapping objects can also be detected.

The SSD model uses a base network (backbone) for classification tasks, and adds several convolutional layers to the end of the network that progressively decrease in size. These convolutional layers help in identifying objects of different aspect ratios. The training is performed end-to-end, using the bounding box locations of each object in the image as input. The training loss used is a combination of localization loss as well as confidence loss. The model architecture of SSD is illustrated in Figure 4-6. The backbone neural net used is the VGG16 net.

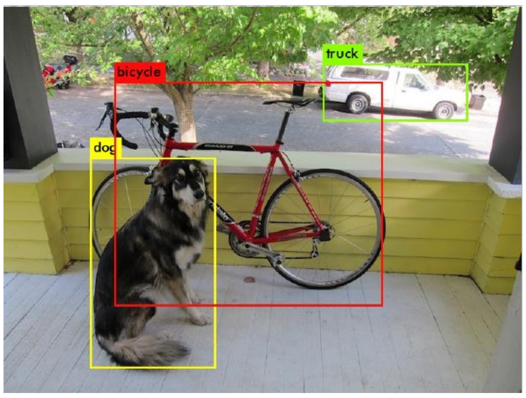

Figure 4-5 An example of the result of running object detection over an image. The interesting objects are enclosed in bounding boxes, with labels indicating the class of object. Picture credits: https://pjreddie.com/darknet/yolo/

The authors report an mAP 81.6% over the PASCAL VOC2007 dataset, and 80.0% over the PASCAL VOC2012 dataset, which is higher than both YOLO and Faster-RCNN. At the same time, it is the fastest DCNN object detector at the time of publication, running at 59FPS on a NVidia Titan GPU.

For this project, keeping in mind that low computational power is desired, an alternative backbone network – the MobileNetv2 model is used in combination with the SSD as the object detection model.

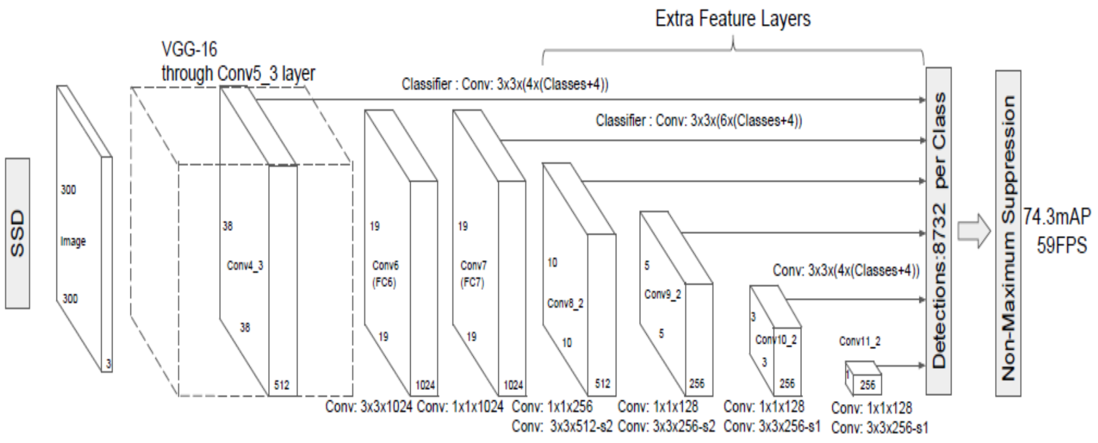

Figure 4-6 The architecture of the SSD object detection framework. Picture Credits: [14]

4.3.1.2 MobileNetv2

The MobileNetV2 [57] architecture uses the concept of depth-wise separable convolutions to reduce computational costs, along with several other improvements to reduce the number of computations. One such technique is the use of inverted residual blocks, where two bottleneck layers are connected through a shortcut link, with the advantage being that the new method is more memory efficient. As a result, MobileNet architecture is well suited for low powered and memory constrained devices such as mobile phones and micro-controllers. The paper also describes a combination of the MobileNetV2-SSD architectures, using a variant of SSD called SSDLite. The resultant network achieves a 22.1% mAP over the COCO dataset, which is close to the 23.2mAP reported for the SSD network on the same dataset. So effectively, the MobileNetV2 architecture has reduced memory and compute requirements while maintaining the accuracy of object detection.

4.3.1.3 Summary

MobileNetV2-SSD network is the clear choice among the explored possibilities, since it meets all needed requirements of the system – high accuracy, low power and low computations. A summary of the object detection methods considered, along with their advantages and disadvantages are shown in Table 4-1.

4.3.2 Feature Extraction and Matching

The features of the object enclosed in the bounding box need to be extracted and matched with previously known features. The feature extraction can be done in multiple ways: conventional methods such as SIFT and SURF, Haar Wavelets etc. were used for their robustness and invariance even under slight changes in illumination or under object translation. However, DCNNs are once again the preferred choice here, since typical operations of the neural networks include extracting features of the object in the image. As the image flows through the network, each layer progressively extract more complex and abstract features from the image. The final layer, the classification layer, contains only the information most closely related to the class or species of the object.

Since the final layers contain more and more class specific features, it becomes difficult to differentiate between two individuals of the same class once the image has passed through to the upper layers. It is due to this reason that typically lower layers of the DCNN are used as feature extractors. DCNNs which have been pre-trained over large datasets and are capable of classifying multiple categories of objects in particular are known to extract better features.

Approach	Pros	Cons
YOLO	• Has multiple variants that trade-off accuracy and speed, allowing the user to select the correct model on an application basis	• Comparatively uses more number of hyper parameters • The faster variants are significantly less accurate
SSD	• Faster than YOLO and Faster-RCNN, and also more accurate • The classification backbone can be replaced with a different network	• Although it works fast on GPUs, CPU speeds are reduced
MobileNetV2-SSD	• Networks are optimized for low-compute resources • Good balance of accuracy and computation need	• Has a slightly reduced accuracy

Table 4-1 Summary of the Object Detection methods considered

Due to the reasons mentioned above, the feature extraction is done using a DCNN pre-trained over the ImageNet dataset. Three models have been considered to be used as feature extractors: AlexNet, GoogLeNet and ResNet. These are some of the most popular DCNNs in use today, especially for feature extraction.

4.3.2.1 *Summary*

Among the three different networks, it is difficult to determine beforehand which of them can act as the best feature extractor. In addition, finding out which layer in the network has the best features to maximize intra class variance is a challenge in itself. A grid-search like approach has been used in this project in order to find the best model and the best layer within the model for the feature extraction step. The complete mechanism used to determine this is described in the Next chapter. The summary of the three DCNNs evaluated in this project is provided below.

Model	Pros	Cons
AlexNet	• Easy to understand, has a simple structure • Less number of layers implies faster prediction time	• Has comparatively very large number of hyper-parameters, which would require more computational power
GoogLeNet	• Typically used for feature extraction, and is known to have good results • Model built for optimizing number of hyper-parameters, so it is computation friendly	• It's nonlinear architecture makes it difficult to understand how the layers react to different features
ResNet50	• Provides a good balance of accuracy and speed	• Not a typical feature extraction network, but widely used for classification problems

Table 4-2 Summary of the DCNN models and their advantages and disadvantages

4.3.3 Extracting the correct set of features

Each animal used in the study have different distinguishing features: tigers have unique stripes; jaguars are identified using their spot patterns, whereas elephants are recognized mostly by their facial features such as shape of the ears. The question arises as to how a single layer in the network can find these distinct features for such varied requirements. The answer is fairly simple – there is no need to use a single layer or even a single model in order to find the individual.

Once the class of animal has been found from the result of the object detection step, each class can use a different model or a different layer from the same model to find the best feature vector. As long as the same model and layer is used for a particular class, the hypothesis presented here is that the process will be able to extract the most distinguishing feature for individual within that species. The general idea of what is mentioned here is illustrated in Figure 4-7.

4.3.4 Identity Association

The penultimate step in the object tracking process is that of identity association. The features for each object have been extracted, and now they are matched with known features and a new id or a previously existing id is assigned to each bounding box.

Identity association is an open-set problem, which means that the classifier used for the association process must be able to determine if a particular individual has not been seen before i.e. the features for the animal fall under the "None of the above" category, and this must indicate to the system that the individual in question has not been encountered before. Two alternatives being considered for this step are the use of clustering, or the use of the Support Vector Machines. Both have their own challenges and advantages, and are explored below.

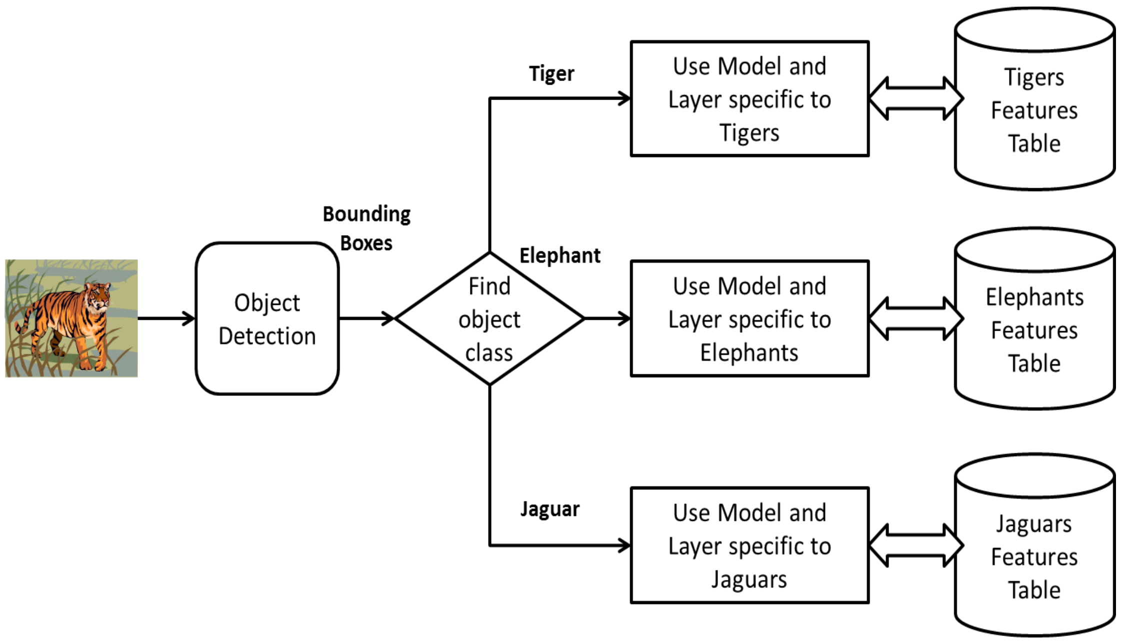

Figure 4-7 Use of different models/layers for feature extraction for different animal species.

4.3.4.1 Clustering

The fact that identity association is an open-set problem means that unsupervised learning methods such as clustering are a potentially good candidate. Visually, if the feature vector were represented as a point in an n-dimensional space, clustering results would be akin to grouping the n-d points of each individual together. Feature vectors extracted from new images would be close to the other points in a particular cluster, and the identity associated with cluster is provided for the new individual as well. However, there arises the problem of choosing two factors: the number of clusters, and the distance metric. The feature vector is quite possibly a very high dimensional vector. This would mean that typical distance measures like Euclidean distance or City-Block distance may not capture the intra-class distance correctly, leading to the formation of poor clusters.

Secondly, choosing the number of clusters has a major impact on the clustering results. It is not possible to select before-hand the number of clusters that the solution should use, given that the number of clusters should ideally be equal to the number of identities, and that the number of identities is not known, the clustering mechanism would be required to compute the number of clusters dynamically, constantly changing it to adapt to new identities.

For the current phase of the project, clustering is not being considered for identity association due to the challenges listed above.

4.3.4.2 Support Vector Machines

SVMs [25] are a popular class of learning systems for classification. Although originally introduced in the context of binary classification, SVMs are capable of solving multi-class problems as well.

The proposed method for identity association uses SVMs to classify the feature vector into one of the categories of known identities. Open Set SVMs are also in use today, that can output an “unknown” class for the input vector. However, for the initial phase of the project, closed set SVMs will be used to find the identity of an individual.

4.3.4.3 Summary

A brief summary of the two identity association methods are tabulated in Table 4-3

Method	**Pros**	**Cons**
Clustering	• An unsupervised learning approach, so it fits rather well to this problem	• Number of clusters not known before hand • Distance metric for intra-class variations is difficult to estimate
SVM	• Widely used approach in classification	• Depending on the type of Kernel used, could lead to intensive computational requirement

Table 4-3 Summary of identity association methods

5 IMPLEMENTATION DETAILS

This chapter describes the details of the implementation for the animal identification part of the system. The tracking of motion and notifications are part the next phase of the project, and are not described here.

The software and tools used are described in 5.1. Different modules used in the project are described in 5.2. The procedure used to find the best layer for feature extraction is described in 5.3.

5.1 Software and Tools

The project runs on a Laptop that has an Intel Core i3 Processor, and is configured with Ubuntu 18.04, 64bit. The system has 8GB RAM. The entire project has been developed in Python 3.6. The tools and libraries used in python for various tasks are summarized in Table 5-1.

Task	Library / Framework	Version
Reading Images, Resizing Images	OpenCV	4.1
Classifiers for Feature Extraction	OpenCV	4.1
Object Detection Framework	Tensorflow	2.0
MongoDB Interfacing	Pymongo	3.9
Vector Operations	Numpy	1.17
PCA, SVD transformations	Scikit (Sklearn)	2.0
Support Vector Machines	Scikit (Sklearn)	2.0

Table 5-1 Python libraries and their version for the various tasks used in the project

5.2 Modules

5.2.1 Object Detection

The object detection framework used is Tensorflow, which is a popular machine learning framework. As described in 4.3.1, the object detection model used is MobileNetV2-SSD, due to its fast processing rate and relatively high accuracy. The model is pre-trained over the *OpenImages* [58] database, and is readily downloadable from the Tensorflow Github Model Zoo. OpenImages is a very large image database containing nearly 9Milliion images spread across 600 categories, and contains annotations for classification, object detection and visual relationship detection. All three species targeted in this study, jaguars, tigers and elephants are all present in the OpenImages dataset. It is thus convenient to directly use the available pre-trained model for object detection. The input image size for the detection framework is 300 x 300 x 3.

5.2.2 Feature Extraction

Multiple choices exist for DCNNs for feature extraction. Three models considered here are AlexNet [6], GoogLeNet [18] and ResNet50 [19] models. Each of these models are pre-trained over the ImageNet database, which has nearly 1Million images spread over 1000 different categories. The model weights are downloadable from various different sources, but not all three together are available for the Tensorflow library. As a consequence, OpenCV's DNN module has been used for feature extraction process, since all the three models officially export OpenCV compatible model metadata and weights. As described in 4.3.2, different layers of the same model or different models altogether can be used to find the feature vector for different species of animals. Each model has different input dimensions for the image, which are summarized in Table 5-2.

5.2.3 Identity Association

The SciKitLearn library's SVM module is used to predict the identity of the individual in the image. The SVM is trained over a database consisting of image-id mappings for animals of all the three species. As before, there could be three different SVM models for each of species, so that the feature vector mappings need not be same for all categories under test. SciKitLearn's library doesn't have an OpenSet SVM classifier, which would have been the preferred approach for this step. Since there is no official python version of an OpenSet SVM classifier that's publicly available, the current phase of the project uses the default closed-set SVM for the classification.

The SkLearn's *TruncatedSVD* pre-processing step is applied to reduce the dimensionality of the input feature vector if required. The feature vector is flattened to a one dimensional vector, and fed into the SVM. The parameters for the SVM, such as the kernel to use, value of gamma, error rate (C) are found empirically. The procedure to find these is described in the next section.

5.3 Finding the best feature extractor

One of the characteristics of CNNs is their inability to explain what features are captured in the model and how these features help classification. There are several methods to visualize the weights of the convolutional layers [59], and some run the network layers in reverse to understand how the network stores information in its layers [60]. However, the weights saved and the features extracted still remain mostly un-understandable to humans. Given the lack of such understanding, the approach used in this thesis is to empirically find the best layer for feature extraction. The process followed to find the layer and model for each category of animal is illustrated in Figure 5-1. The various parameters considered is summarized in Table 5-3. The pseudo code for the process is shown in Figure 5-2.

The features for every layer of every model considered in this process are saved to a database. *MongoDB* is used here for its convenient access and storage functions. Every iteration of the algorithm considers a different set of parameters, and evaluates their effectiveness. The best combination of parameters is saved to disk. The Annotations for each image is an integer identifier, which forms the target for the features extracted from that image. The results of running this evaluation are presented in the next chapter.

Model	Input Image Dimensions
MobileNet-SSD	(300, 300,3)
AlexNet	(227,227,3)
GoogLeNet	(224,224,3)
ResNet50	(224,224,3)

Table 5-2 Input dimensions for the different models used.

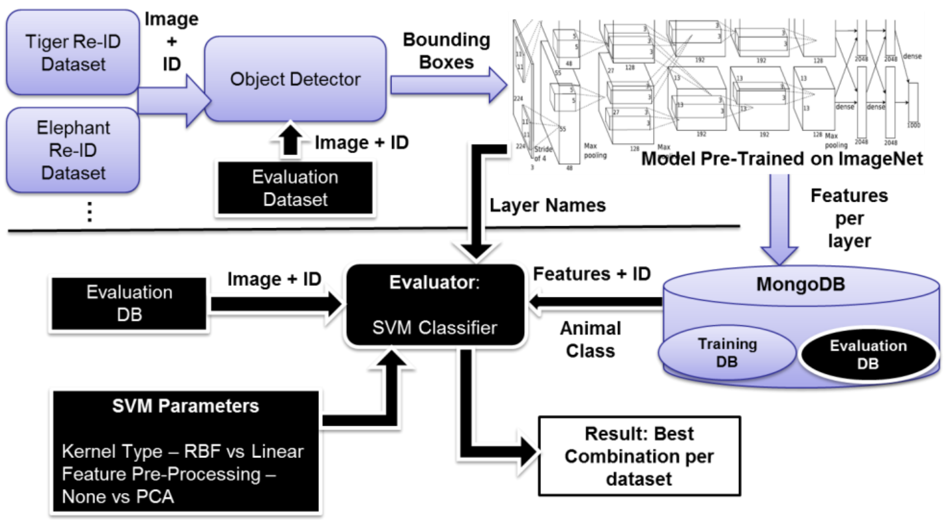

Figure 5-1 The process used to find the best layer and model for optimal feature extraction of each class

Component	Function	Parameters
Model	Extraction of Feature Vectors	AlexNet, GoogLeNet, ResNet50
Dataset	Contains Images and their labels	Jaguars, Elephants, Tigers
Evaluator	SVM	-
Pre-Processing	Reduce dimensionality if needed	None, PCA
SVM Parameters	The Kernel Type	RBF, Linear
SVM Parameters	Gamma Values	Scale, floating points in range [1.0, 2^{16}]

Table 5-3 Summary of parameters tested to find optimal feature extractor

```
findBestParameterCombination(datasets, classifiers):
  for dataset in datasets, do
    for each model in classifiers, do
      for each layer in model, do

        for each image in the evaluation set, do
          features = read features from database for image from layer
          done;

        evaluationSet = seggregate images randomly into two sets - train and eval
        TransformationType = [None, PCA]
        KernelType = [RBF, Linear]

        bestModel = initialize a blank model
        for each combination of parameters in (TransformationTypes, KernelType), do
          Fit SVM using targets for each feature vector in training set
          score = Evaluate SVM over a different evaluation set of images
          if score > bestModel, do
            save model to disk, along with all necessary parameters
            remove previously saved models
            bestModel = current combination of parameters
    output (dataset, best)
```

Figure 5-2 Pseudo-code for finding the best model for identification per dataset.

6 Intermediate Results and Discussion

This chapter presents the intermediate results found for the procedures elaborated in the previous section, and also describes the datasets used for the project, and the evaluation metrics considered. The results presented here are intended only as a baseline, and will be further worked upon to improve them in the next phase of the project.

6.1 Datasets

There are three datasets that are used in this phase of the project – one each for tigers, jaguars and elephants. The tigers dataset was acquired from the Amur Tiger Re-Identification challenge [40]. It contains over 100 individuals and nearly 1800 re-id related images. The dataset on elephants was acquired on request from the Elephant Listening Project [61]. The dataset contains images of 276 individuals, and has over 2000 images. The image set of Jaguars was provided on request from Dr. Marcella Kelly, Department of Fish and Wildlife Conservation, Virginia Tech. This dataset has over 1000 images of 50 different individuals.

The datasets have been summarized in Table 6-1. A sample set of images for each dataset are shown in Figure 6-1, Figure 6-2 and Figure 6-3. Figure 6-4 shows the cropped images of a randomly selected individual Jaguar and Tiger to show the diversities and challenges involved in id prediction.

Dataset	Number of Images	Number of Individuals
Elephants	2078	276
Tigers	3392	107
Jaguars	1083	50

Table 6-1 Summary of each dataset, with the number of images and number of individuals for each category.

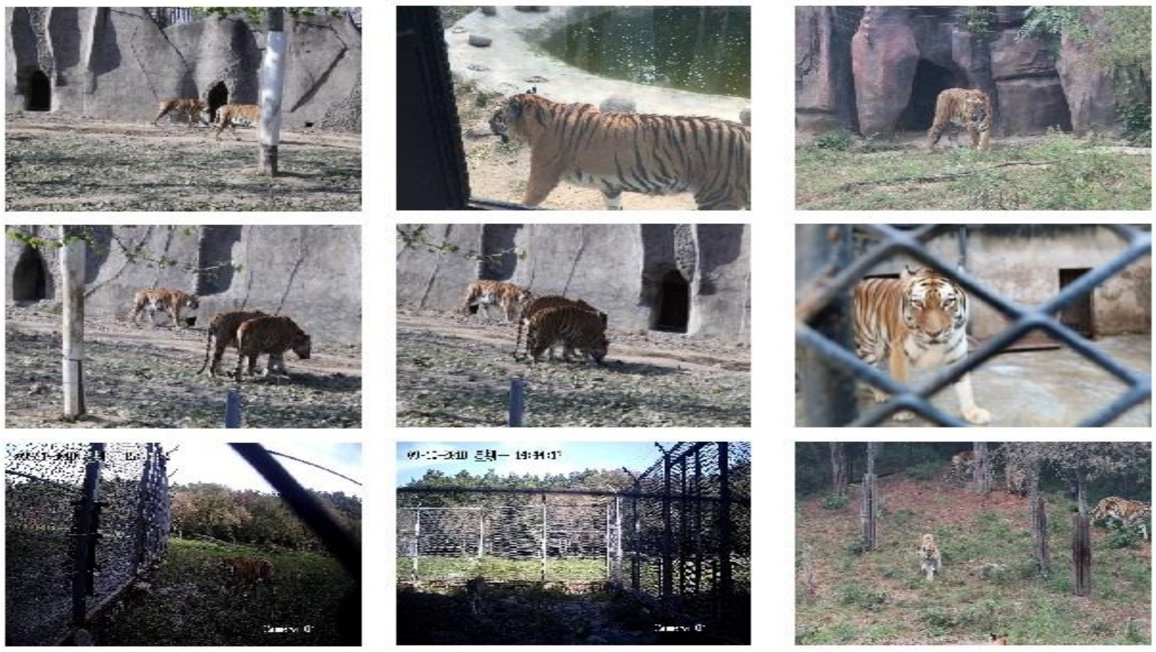

Figure 6-1 A sample set of images from the Tigers Data Set. The images show tigers in various scenes, and visible at various scales.

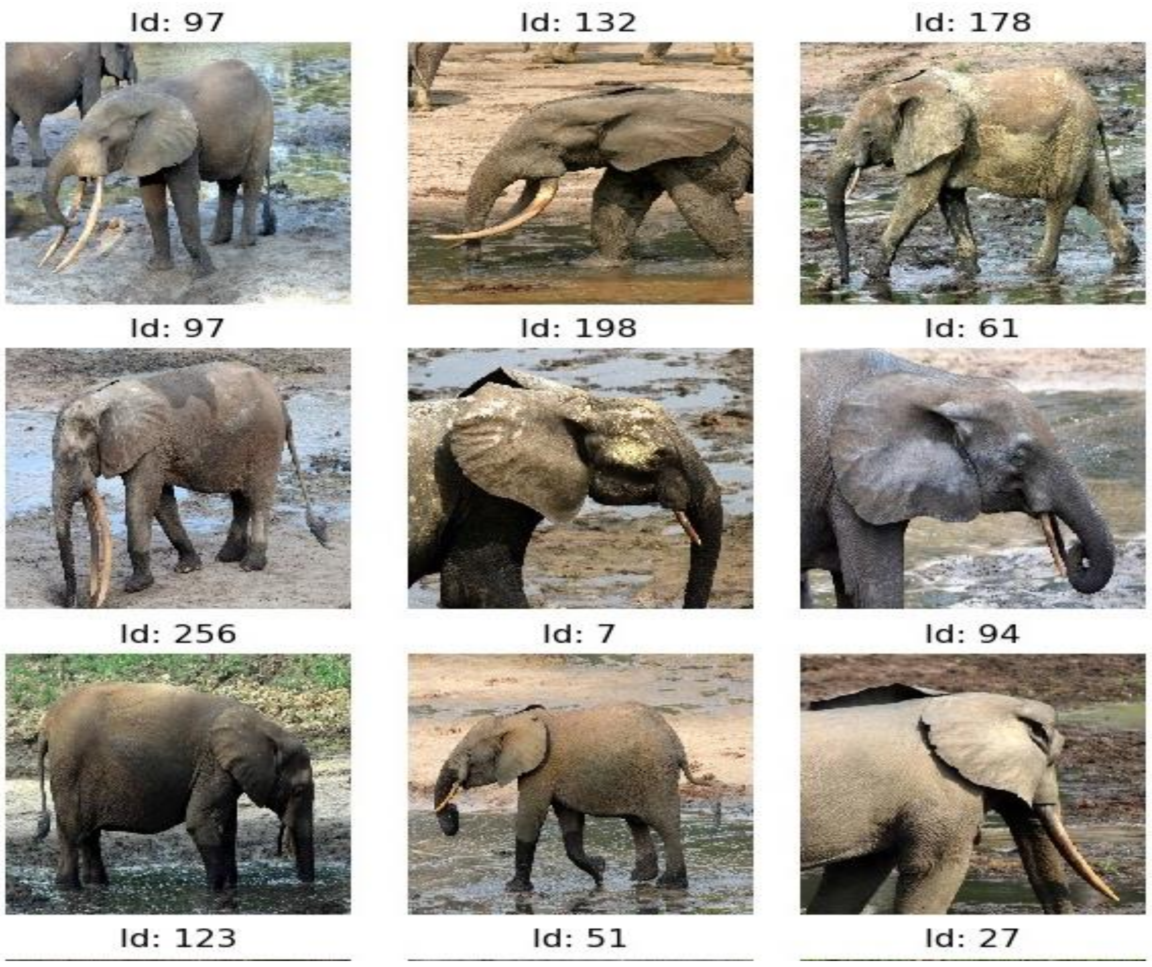

Figure 6-2 A sample set of images from the Elephants Data Set. The images show elephants in different backgrounds. The identifier of each element appears directly above it in the image.

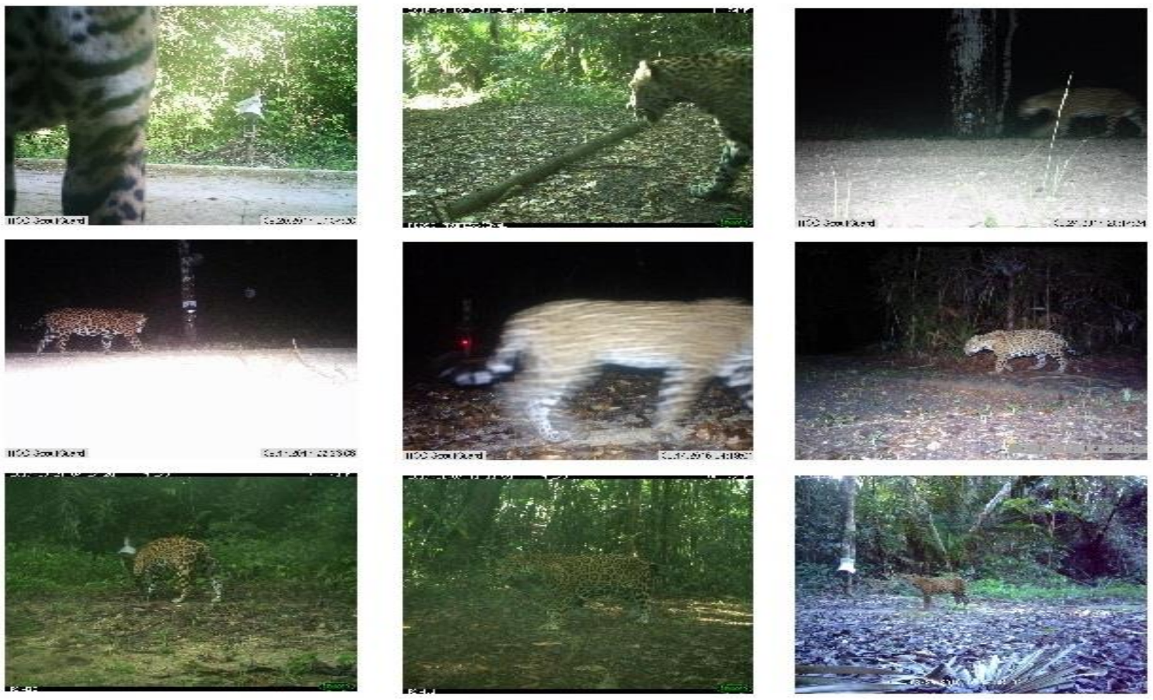

Figure 6-3 A sample set of images from the Jaguars Data Set. The images show jaguars in various scenes, and visible at various scales, some too close to the camera, and some too far away.

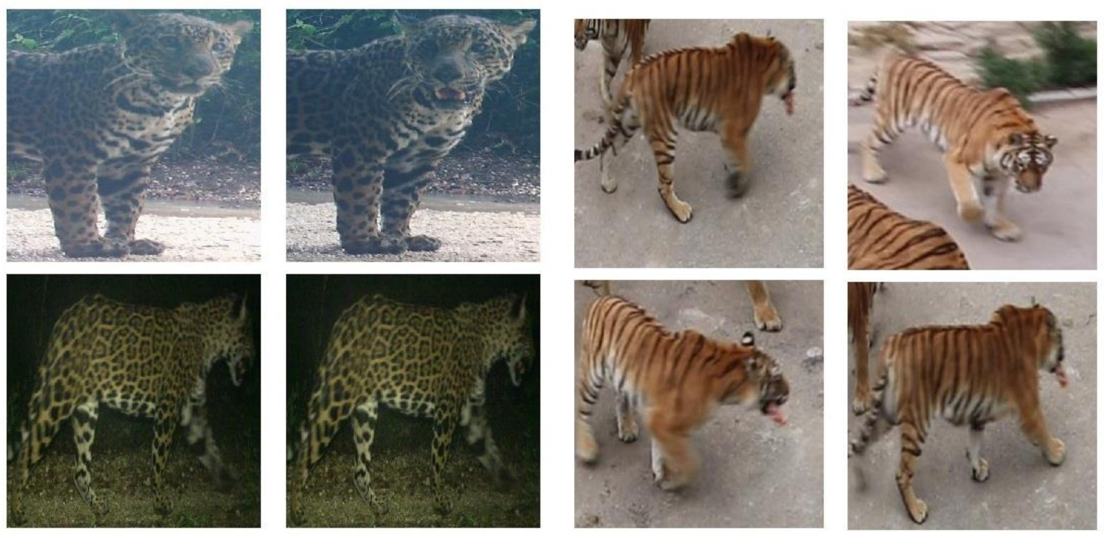
Figure 6-4 Some images of an individual Jaguar and Tiger. The animal can be seen in multiple poses.

6.1.1 Annotations

Each of the three datasets are further split into two categories – a re-identification dataset, and a detection dataset. The re-identification dataset consists of cropped images of the animals and is helpful to directly run identity association tests. The annotations for the re-id dataset are simply an identifier for the animal(s) in the image. This can be specified as simple text entry, or even as part of the image name. The detection dataset contains the actual images captured through camera traps and other field photography techniques. The annotations for this would require to be a bounding box surrounding each animal visible in the image. The bounding box is annotated as two (x,y) points, representing the top-left and the bottom-right points of the rectangle placed over the animal. This is the PASCAL VOC annotation format, and the annotations are all in XML.

6.1.2 Distribution of Images

It is important that there are no biases present in the data before it is used for training. This however, is very unlikely. The distributions for three different datasets that show the number of images per individual plotted against the number of individuals having as many images in Figure 6-5. Ideally, this should be only one bar, showing every individual has equal number of images. Having large number of images for one or two individuals biases the data and would cause the predictions of the classifier to lean towards the individual that has higher number of images.

Figure 6-5 shows the imbalance in the distribution of number of images. The tigers dataset has one individual for whom the number of images is close to 100. The case is worse in the Jaguars dataset, where a single individual has more than 170 images. This constitutes bias in the dataset, and has significant impact on the Id predictions. The elephants dataset is relatively more balanced.

6.2 Performance Metrics

There are two performance metrics used to evaluate the effectiveness of the system proposed in this thesis – accuracy of identification, and time taken for identification.

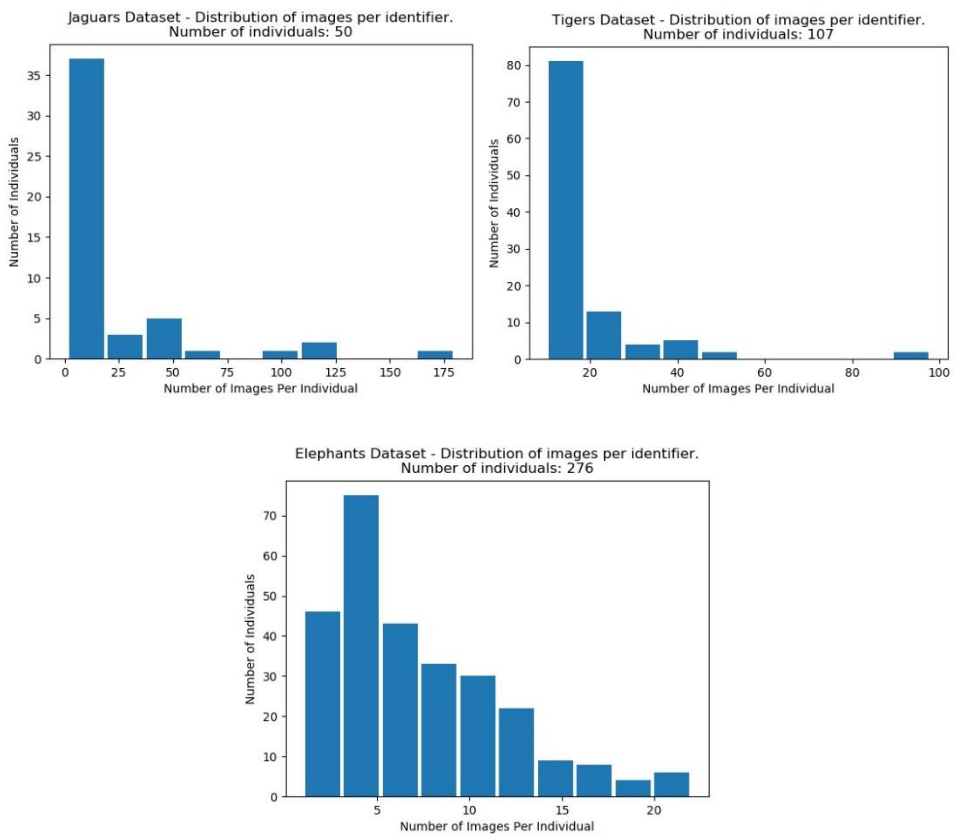

Figure 6-5 Distribution of image samples per subject for each of the three datasets. The X-Axis shows the number of images per individual, and the Y-Axis shows the number of individuals that have the corresponding number of images. For example, in the Elephants dataset, there are close to 75 individuals for whom the number of images per individual is 3-4.

6.2.1 Accuracy of Detection

Detection accuracy is the likelihood of finding the presence of the animal of interest within the image, locating it within the image and correctly categorizing the object to its ground-truth class. The predicted bounding box coordinates need to overlap as much as possible to the ground-truth. The detection of the object and classification is fairly straight forward – the model should correctly find and classify all the animals in the image. This is a binary, hit-or-miss evaluation metric. However, it is slightly more challenging to evaluate the effectiveness of find the bounding box location, since even if the model correctly predicts that there is one animal in the image, if the placement of the bounding box is wrong, then the detection should still be considered as failed.

A general metric used in this regard, the Intersection-over-Union (IOU), measures the overlap between the predicted bounding box, and the ground-truth bounding box, and evaluates it to a number in range [0,1] – where 0 signifies the two boxes do not intersect, and 1 indicates a perfect overlap. An IOU of 0.5 is the success criterion; so if the predicted bounding box overlaps 50% or more with the ground truth, it is considered to be a success.

6.2.2 Accuracy of Identification

The accuracy of detection captures the success rate of the system in identifying two images of the same individual. It denotes how likely it is for the system to associate the same id for the same individual, given a set of images. This should be as close to 100% as possible in theory, but in practice, since additional information can be used to infer the identity of the animal (such as spatial and temporal information of the subject), this need not be as high. Since only closed set identification is being used for the current phase, the identification of an individual is considered a success if the id predicted by the model matches with the ground truth identifier of the animal in the image.

6.2.3 Time

Time is one of the more important metrics since the system is planned to be used for real time and near-real-time applications. There are multiple steps in the identification process, and the target is to achieve a frame rate of around 4-5 FPS, which would mean that the system can take up to 250ms to associate an identity to the detected animal, starting from the time of image capture.

6.3 Results

The results of running the model over each dataset are reported here. All three evaluation metrics, accuracy of detection, accuracy of identification and time taken are reported separately for each dataset.

6.3.1 Detection Accuracy

The accuracy of detection is summarized in Table 6-2. The metrics are reported for two categories of each dataset, namely the detection dataset and the re-id dataset. The model employed here is the MobileNetV2-SSD. The reported results are run over all images available in the dataset, with no additional training applied to the model. An IOU > 0.5 is considered to be a successful classification. A few sample results for each dataset, over the two categories are shown in Figure 6-7, Figure 6-6, Figure 6-8, Figure 6-10 and Figure 6-9. Both successes as well as failures are shown for each dataset.

The results show a fair accuracy for the elephant dataset, but it is poor for the remainder of the datasets. The detection dataset is a much better indication of the success rate of the model, since in the application, images belonging to the detection category are much more likely to occur than those in the re-id category. It is to be noted that the model has not been trained over any specific dataset, or the datasets used in this project. As a result, the detection accuracy is fairly low. This can be improved by transfer learning, which will be carried out in the next phase.

Dataset	**Category**	**Number of Images**	**Accuracy (%)**
Tigers	Detection	2762	13.0
	Re-Id	3392	52.12
Jaguars	Detection	1188	7.57
	Re-Id	1083	17.65
Elephants	Detection / Re-Id	2078	87.1

Table 6-2 Accuracy of detection for different categories of each dataset. There is no separate detection / re-id dataset for elephants.

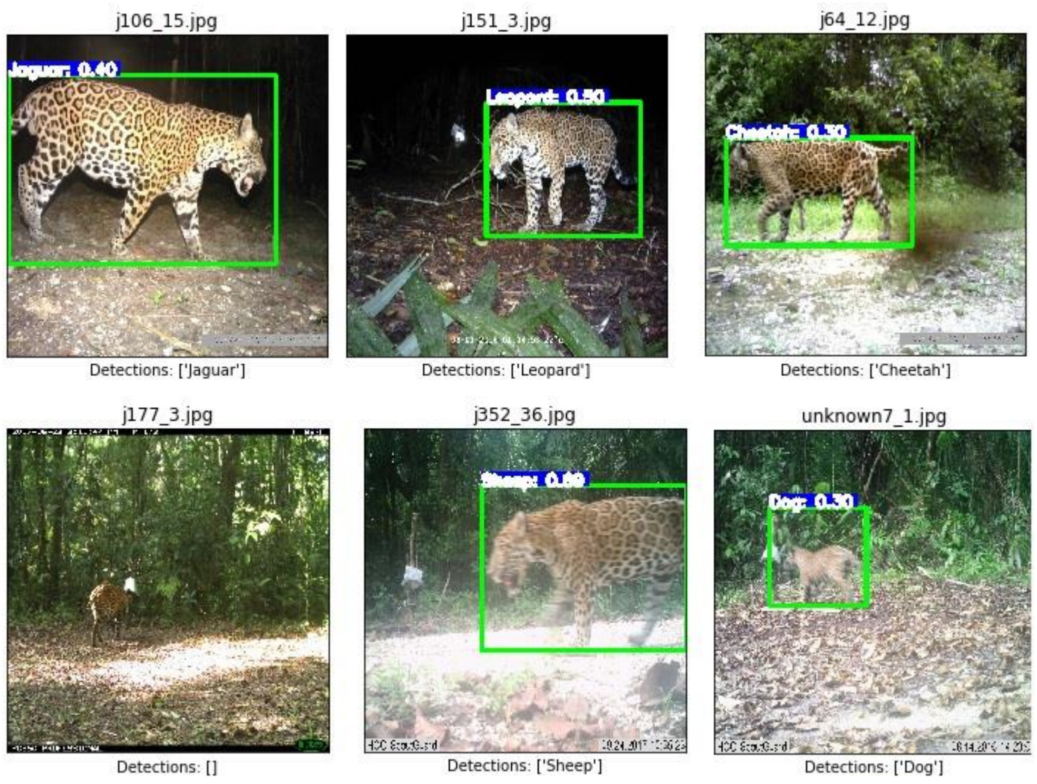

Figure 6-6 Object Detection on the Jaguar Detection dataset, the figure shows both successful and failed results. Jaguar, Cheetah and Leopard are treated as synonyms.

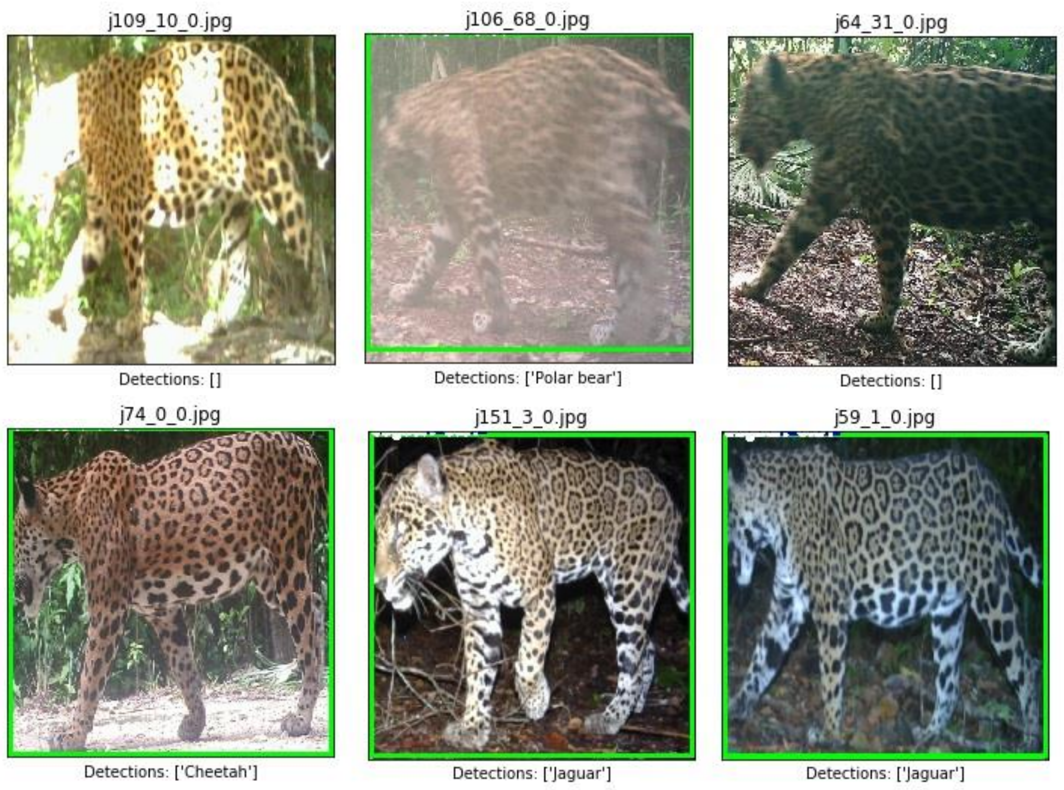

Figure 6-7 Object Detection on the Jaguar Re-Id Dataset, showing sample successful and failed results. Jaguar, Leopard and Cheetah are treated as synonyms here.

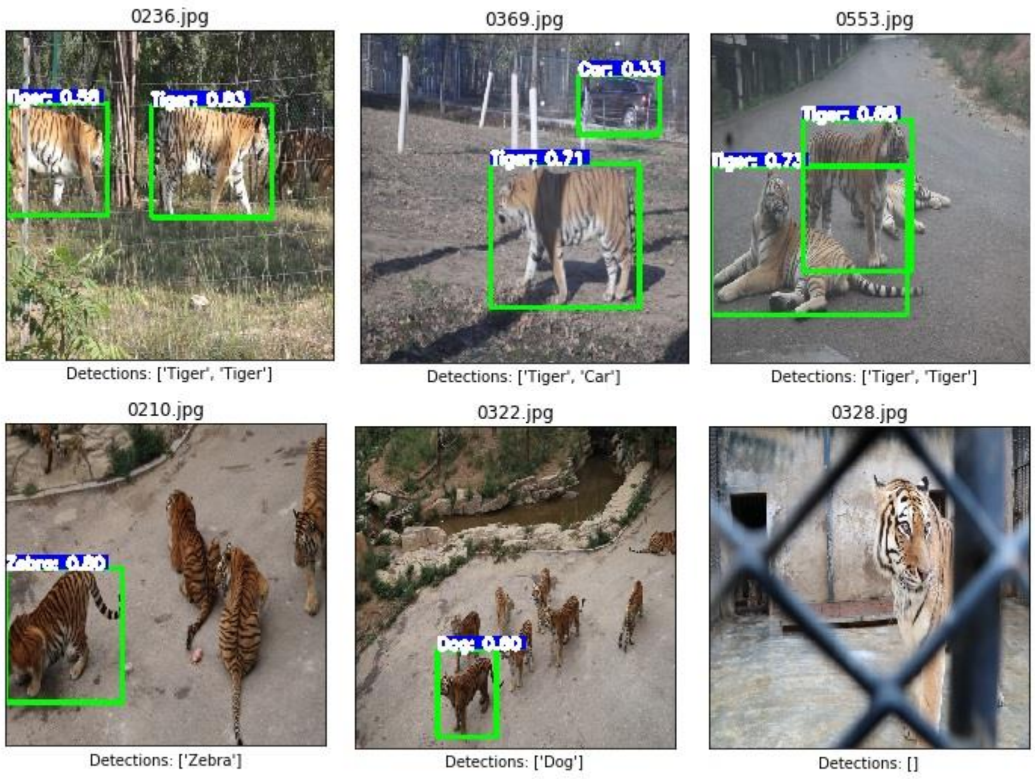

Figure 6-8 Object detection results over the Tigers Detection Dataset. The figure shows both successful and failed results.

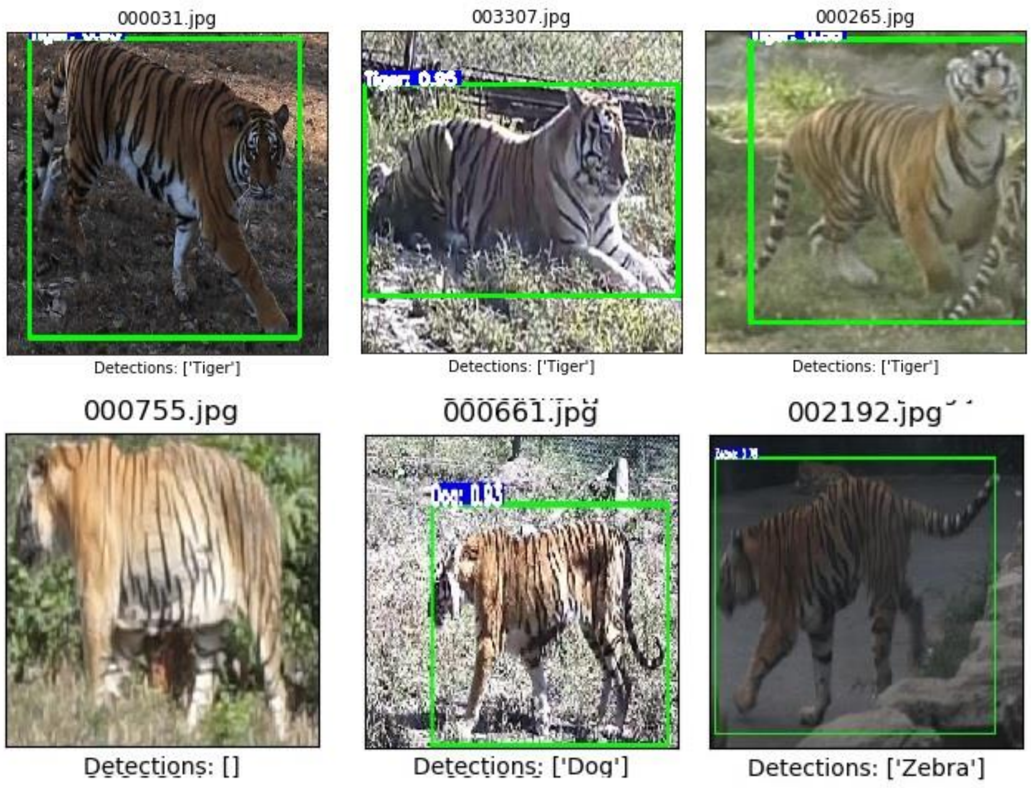

Figure 6-9 Results of running Object Detection on the Tigers Re-ID dataset. The figure shows failed and successful results.

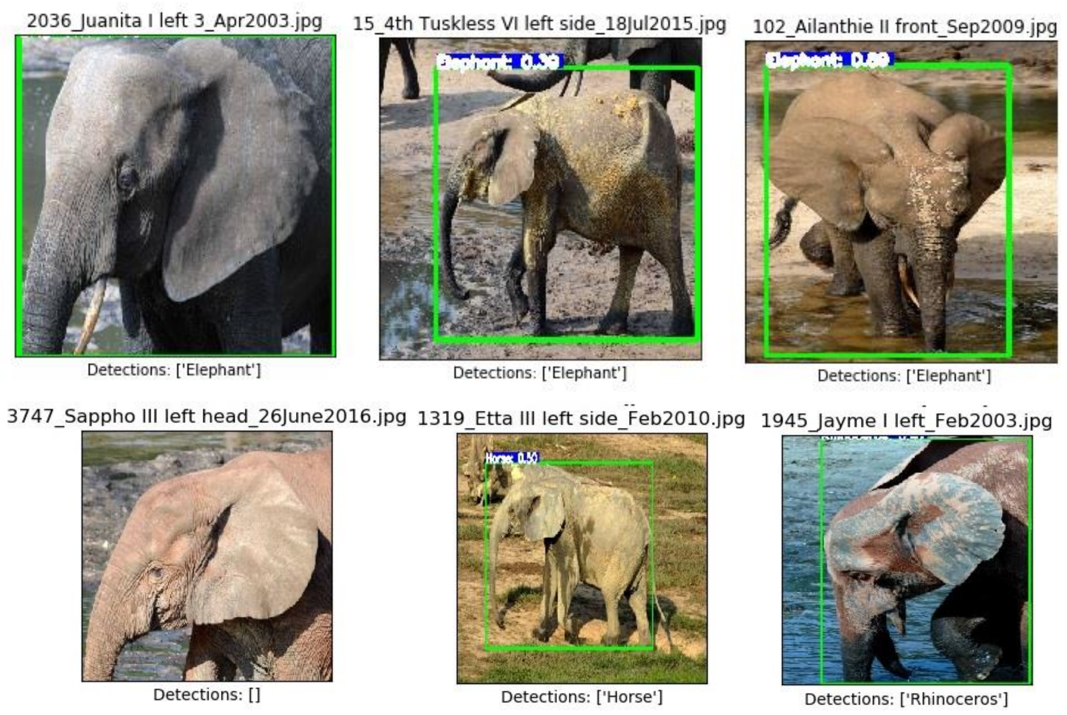
2036_Juanita I left 3_Apr2003.jpg
Detections: ['Elephant']
15_4th Tuskless VI left side_18Jul2015.jpg
Elephant: 0.39
Detections: ['Elephant']
102_Ailanthie II front_Sep2009.jpg
Elephant: 0.89
Detections: ['Elephant']
3747_Sappho III left head_26June2016.jpg
Detections: []
1319_Etta III left side_Feb2010.jpg
Horse: 0.50
Detections: ['Horse']
1945_Jayme I left_Feb2003.jpg
Detections: ['Rhinoceros']

Figure 6-10 Object Detection results over the Elephants dataset, showing both successful and failed detections.

Dataset	No. of Images / Individuals	Model	Layer	Kernel	Pre-processing	Accuracy over validation set	Accuracy over Test Set
Tigers	600/107	AlexNet	Relu6	Linear	PCA	78.9	85.4
Jaguars	600/46	AlexNet	FC7	Linear	PCA	44.5	60.7
Elephants	792/264	AlexNet	FC7	Linear	PCA	5.4	31.8

Table 6-3 Results of the highest accuracy found after running grid-search over the parameters considered.

6.3.2 Identification Accuracy

This section describes the result of the grid-search like approach described in 5.3 to find the best combination of parameters that leads to a model with high accuracy for identification. The metrics reported here will serve as a baseline for the next phase of the project, where the model is fine-tuned for better accuracy.

The best metrics achieved using the methodology proposed in this study are presented in Table 6-3. There were approximately 600 training images used for each dataset, with any individuals having less than 5 samples were removed. When reducing the number of images, they were first reduced from those individuals that had abundant number of training samples, while those with fewer samples were not penalized. The test accuracy was calculated using the entire dataset.

It is to be noted that in all three cases, the test accuracy is higher than the validation accuracy. This indicates that although the validation accuracy is low, the model generalizes well to new images. The most notable statistic here is the accuracy for elephants – it's around 5%, which is extremely poor. The main reasons for this low score include the diversity of the dataset. It contains elephants in various states, such as wet, covered in mud or dry. This affects the color of the elephant, and might be one of the reasons for the lower accuracy. This can be reconfirmed from Figure 6-13, where it shows

9 samples for the same individual, where each image contains the elephant in a distinct viewing angle and condition. Results of identification are illustrated in Figure 6-11, Figure 6-12 and Figure 6-13.

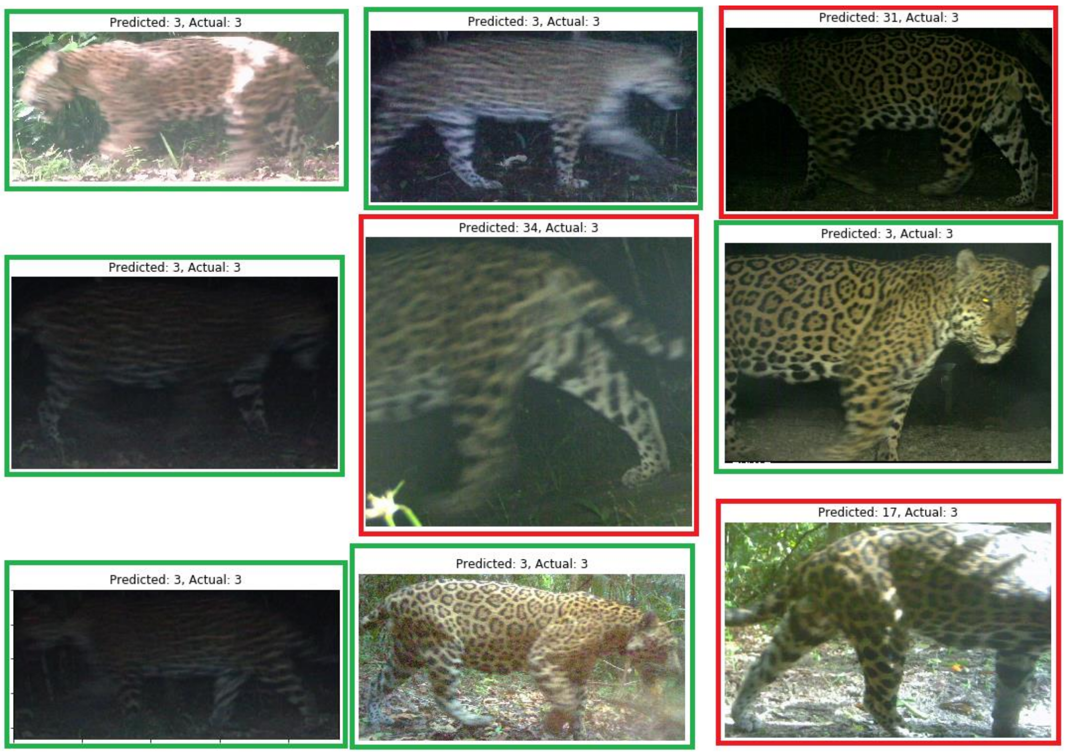

Figure 6-11 Results of running SVM Classifier on all images of Jaguar with Id 3. Green outlines indicate successful cases, and red shows failed ones. The accuracy for this identity is 66.7%.

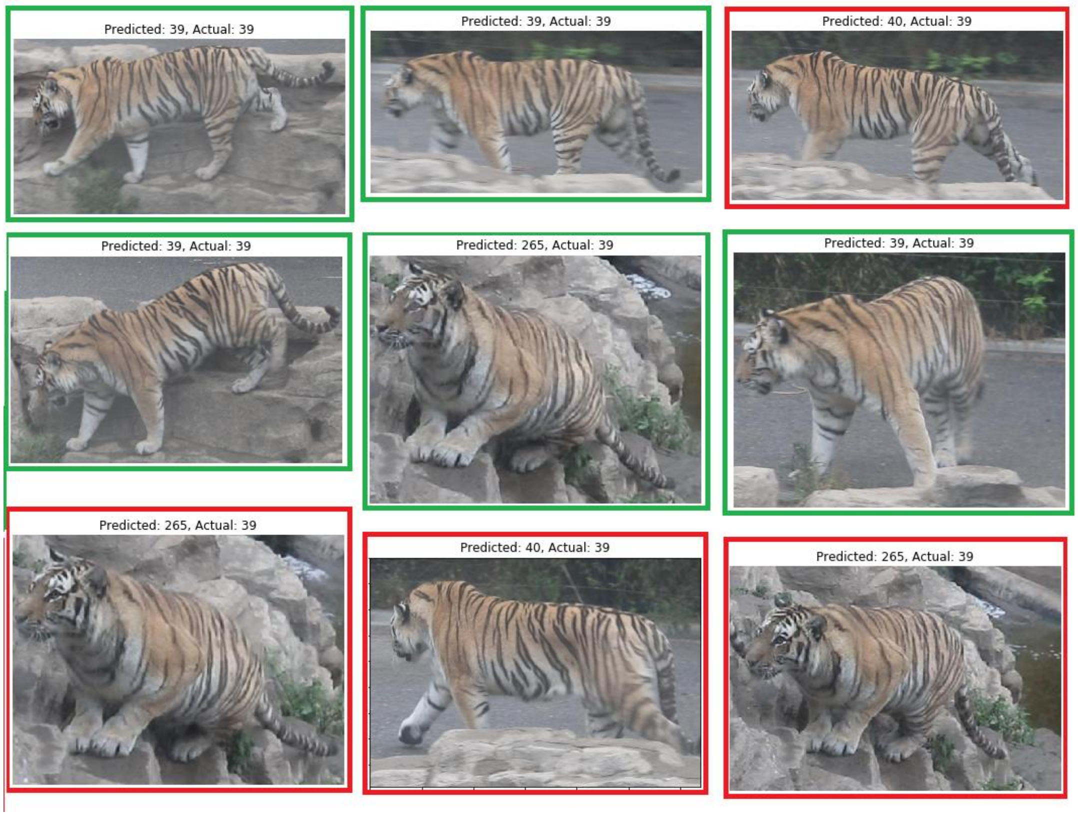
Predicted: 39, Actual: 39
Predicted: 39, Actual: 39
Predicted: 40, Actual: 39
Predicted: 39, Actual: 39
Predicted: 265, Actual: 39
Predicted: 39, Actual: 39
Predicted: 265, Actual: 39
Predicted: 40, Actual: 39
Predicted: 265, Actual: 39

Figure 6-12 Results of running SVM Classifier on all images of Tiger with Id 39. Green outlines indicate successful cases, and red shows failed ones. The accuracy for this identity is 44.9%.

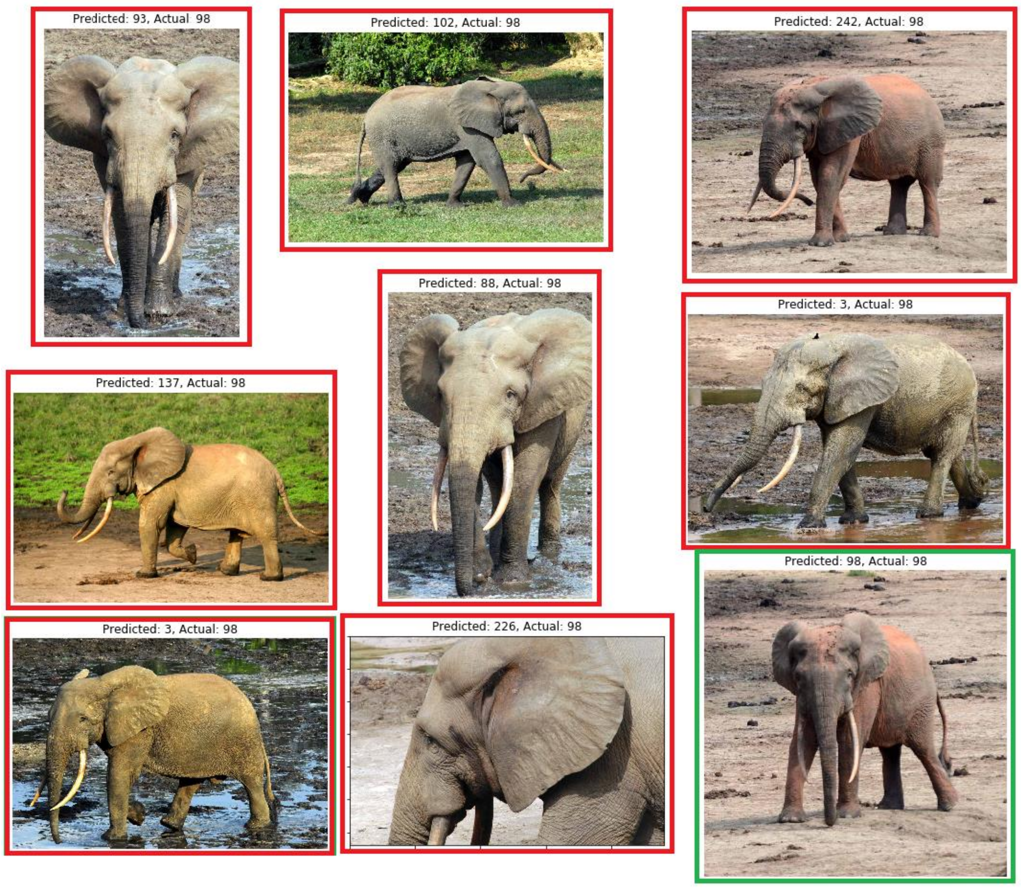
Predicted: 93, Actual 98
Predicted: 102, Actual: 98
Predicted: 242, Actual: 98
Predicted: 88, Actual: 98
Predicted: 3, Actual: 98
Predicted: 137, Actual: 98
Predicted: 98, Actual: 98
Predicted: 3, Actual: 98
Predicted: 226, Actual: 98

Figure 6-13 Results of running SVM Classifier on all images of Elephant with Id 98. Green outlines indicate successful cases, and red shows failed ones. The accuracy for this identity is 11.1%.

6.3.2.1 Removing Bias

As pointed out before, almost all the three datasets contains few individuals for whom there are abundant sample images, and a few individuals for whom there are hardly any training samples. This constitutes a bias towards the individual that contains large number of samples. An experiment was conducted to study the effect of bias on the accuracy of the model. Using the best model for each dataset as indicated from Table 3, the accuracy variation was conducted by varying two factors:

a) Vary the number of individuals, keeping the number of samples per individual constant
b) Vary the number of samples per individual, keeping number of images per individual constant

The results of these two experiments are illustrated in Figure 6-14 for the tigers dataset. As is evident from the graphs, keeping the number of identities constant at 15, and varying the samples per id cause an increase, and slowly drop off. The number of images with 8 or more samples per id was not sufficient to plot the graph for further values. At 4 samples per id, the validation accuracy reaches close to 90%. On the other hand, keeping the number of samples per id fixed at 5, and varying the number of identities show that at around 20 identities, the accuracy reaches close to 90% again, but slows starts to drop off as the number of identities increases. The model that classifies the identities needs to be fine-tuned to these values.

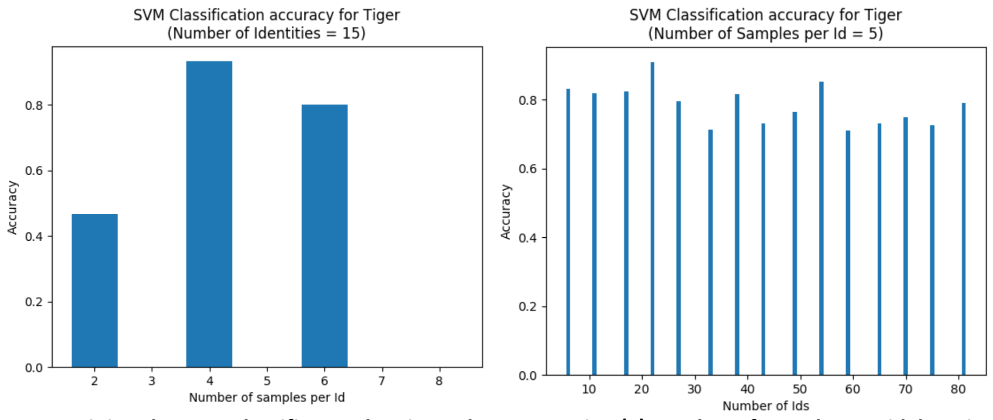

Figure 6-14 Retraining the SVM classifier on the Tigers dataset varying (a) number of samples per id, keeping number of ids constant, and (b) Varying number of identities, keeping number of samples per id constant. The Variations seen from the images show that both factors affect identification accuracy, and attention needs to be paid to how many identities are retained in the DB.

Step	Time Taken (ms)
Object Detection	110
Feature Extraction	125
SVM Prediction	2
Total	**237**

Table 6-4 Time taken for various steps in the identification pipeline, measured over 1000 iterations.

6.3.3 Time Taken

The time taken by each step of the process, measured by running each step separately over 1000 iterations is summarized in Table 6-4. The time taken is well within the pre-defined interval. Since the processor used in this setup is rather slow (Core i3, 6th Gen, 4vCPUs, 2GHz), the expectation is that even when run on an embedded Micro-Controller such as Raspberry Pi (which is a quad-core ARM processor), the time taken shouldn't increase significantly. The timings reported exclude the model load times and the time taken for the very first run, when the model is still warming up. These shouldn't have major effect, since they are one-time occurrences.

6.4 Discussion

6.4.1 Object Detection

The intermediate results obtained here illustrate that readily available pre-trained models are not suitable for direct use in applications. Since the number of class-wise images or their quality is not known, it becomes difficult to quantify the effects of using such models without additional training.

Training the model, however, requires one or more GPUs, since training a DCNN is a very expensive process, often known to run for days or weeks, even with multiple GPUs. In addition, several choices need to be made during the training, such as the use of the loss function, the learning rate, momentum decay, dropout rates and the optimizer used. Typical transfer learning methods would freeze the weights of all but the last few layers, add some layers at the end and then train the weights of only the newly added layers for optimum accuracy. This is the procedure that is intended to be followed here as well, and will be completed as part of the next phase of the project.

6.4.2 Object Identification

There is clearly lot of room for improvement here. The low accuracy rates can be attributed to the following factors:

a) Low detection rates mean that the bounding boxes are not tightly placed over the animal. Some portion of the background will also be extracted along with the features of the animal. More accurate object detection will lead to increased accuracy
b) The classification models used are not sufficiently trained to extract features that discriminate intra class objects
c) Insufficient discriminating features in the image. This appears especially true for elephants, where the accuracy is hardly better than random guessing. Finer details need to be extracted, such as face or ears, to perform the identification.
d) The features being extracted from the pool and convolutional layers are multi-dimensional. They need to be flattened before being used in an SVM. The process of flattening removes all spatial information contained in the feature, thereby largely affecting accuracy
e) The hyper-plane separating the identities exists in a higher dimension than what is being tested for here. This would mean that additional kernel types need to be tested

Alternatively, it is also possible that SVMs are a poor choice of learning model for the identity association problem. Neural Networks could be used instead, since their inherently non-linear mappings can be leveraged to obtain better accuracies. These alternatives will be attempted in the next phase of the project.

7 CONCLUSIONS AND FUTURE WORK

7.1 Conclusions

Although the project is still in its initial stage, some conclusions can be drawn from the work carried out so far. The overall system architecture, along with each component, the modules and their functionalities are clearly defined. Each have a specific role to play in the process, and have been designed to be optimal in their functionalities.

Computationally, the pipeline described in this project meets the requirements quite well. All parts of the pipeline are built such that they consume lower resources, and work fast. The expectation is that even if the same pipeline is moved to an embedded system, there would not be a significant increase in the time taken.

The datasets provided are challenging – a lot of the images contain the animals far away from the camera. There are also some images where there is low lighting, shadows and occlusions, all of which add to the challenge of achieving good detection rates. This is especially true for the Jaguars dataset, where some images are difficult even for humans to locate the animals. The DCNN model needs to be trained further to improve its detection accuracy. Transfer learning needs to be employed so that the number of classes can be reduced. This also helps achieve better identification accuracy, since the objects in the image are bound more tightly leading to better feature extraction.

7.2 Future Work

Although there are still some open questions, a few of the targeted goals have been met. The software implementation is nearing completion, and the work for next phase i.e. tracking using identifiers can begin. The following items are being targeted for the subsequent stage:

a) Improving accuracy of Object Detection through Transfer Learning
b) Use of Open Set identification as opposed to Closed Set Identification used here
c) Improve accuracy of object identification and association
d) Use of IR images along with visible light images.
e) Forming the identification pipeline using IR or thermal images
f) Obtaining the direction of movement through tracking
g) Creating notifications if the movement is found to match with some preset rules
h) Embedding the software on a micro-controller(s)

Acknowledgements

I am thankful to Peter Wrege, Cornell University and Matthias Korschens, University of Jena for their prompt response, and also agreeing to share the Elephants dataset with me.

I am also thankful to Dr. Marcella Kelly and Brogan Holcombe, Virginia Tech, who have agreed to share their jaguar dataset.

I would also like to appreciate Dr. R Krishnan, Dayanand Sagar College of Engineering, for his time, patience and valuable inputs which have helped make the project better.

REFERENCES

[1] "Video Tracking," 18 November 2019. [Online]. Available: https://en.wikipedia.org/wiki/Video_tracking. [Accessed December 2019].

[2] N. Paragios and R. Deriche, "Geodesic active contours and level sets for the detection and tracking of moving objects," *IEEE Transactions on pattern analysis and machine intelligence,* pp. 266--280, 2000.

[3] D. Comaniciu, V. Ramesh and P. Meer, "Real-time tracking of non-rigid objects using mean shift," in *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662),* 2000.

[4] O. Zoidi, A. Tefas and I. Pitas, "Visual object tracking based on local steering kernels and color histograms," *IEEE Transactions on Circuits and Systems for Video Technology,* pp. 870--882, 2012.

[5] F. Online, "Biofencing: A unique yet effective method to tackle human-animal conflict in Uttarakhand!," 19 September 2019. [Online]. Available: https://www.financialexpress.com/lifestyle/science/biofencing-a-unique-yet-effective-method-to-tackle-human-animal-conflict-in-uttarakhand/1710763/. [Accessed December 2019].

[6] A. Krizhevsky, I. Sutskever and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems,* pp. 1097--1105, 2012.

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, 2009.

[8] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014.

[9] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015.

[10] S. Ren, K. He, R. Girshick and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015.

[11] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[12] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[13] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," in *arXiv preprint arXiv:1804.02767*, 2018.

[14] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu and A. C. Berg, "SSD: Single Shot MultiBox Detector," *Lecture Notes in Computer Science,* p. 21–37, 2016.

[15] M. S. Norouzzadeh, A. Nguyen, M. Kosmala, A. Swanson, M. S. Palmer, C. Packer and J. Clune, "Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning," *Proceedings of the National Academy of Sciences,* vol. 115, pp. E5716--E5725, 2018.

[16] A. Swanson, M. Kosmala, C. Lintott, R. Simpson, A. Smith and C. Packer, "Snapshot Serengeti, high-frequency annotated camera trap images of 40 mammalian

species in an African savanna," *Scientific data*, vol. 2, p. 150026, 2015.

[17] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[18] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015.

[19] K. He, X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[20] G. K. Verma and P. Gupta, "Wild animal detection using deep convolutional neural network," in *Proceedings of 2nd international conference on computer vision & image processing*, 2018.

[21] W. Feng, J. Zhang, C. Hu, Y. Wang, Q. Xiang and H. Yan, "A novel saliency detection method for wild animal monitoring images with WMSN," *Journal of Sensors*, vol. 2018, 2018.

[22] S. Matuska, R. Hudec, M. Benco, P. Kamencay and M. Zachariasova, "A novel system for automatic detection and classification of animal," in *2014 ELEKTRO*, 2014.

[23] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, pp. 91-110, 2004.

[24] H. Bay, T. Tuytelaars and L. Van Gool, "Surf: Speeded up robust features," in *European conference on computer vision*, 2006.

[25] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, pp. 273--297, 1995.

[26] Z. Zhang, Z. He, G. Cao and W. Cao, "Animal detection from highly cluttered natural scenes using spatiotemporal object region proposals and patch verification," *IEEE Transactions on Multimedia*, vol. 18, pp. 2079-2092, 2016.

[27] H. Nguyen, S. J. Maclagan, T. D. Nguyen, T. Nguyen, P. Flemons, K. Andrews, E. G. Ritchie and D. Phung, "Animal recognition and identification with deep convolutional neural networks for automated wildlife monitoring," in *2017 IEEE international conference on data science and advanced Analytics (DSAA)*, 2017.

[28] M. J. Wilber, W. J. Scheirer, P. Leitner, B. Heflin, J. Zott, D. Reinke, D. K. Delaney and T. E. Boult, "Animal recognition in the mojave desert: Vision tools for field biologists," in *2013 IEEE Workshop on Applications of Computer Vision (WACV)*, 2013.

[29] S. Schneider, G. W. Taylor, S. Linquist and S. C. Kremer, "Past, present and future approaches using computer vision for animal re-identification from camera trap data," *Methods in Ecology and Evolution*, pp. 461--470, 2019.

[30] A. Pérez-Escudero, J. Vicente-Page, R. C. Hinz, S. Arganda and G. G. De Polavieja, "idTracker: tracking individuals in a group by automatic identification of unmarked animals," *Nature methods*, vol. 11, p. 743, 2014.

[31] A. Rodriguez, H. Zhang, J. Klaminder, T. Brodin, P. L. Andersson and M. Andersson, "ToxTrac: a fast and robust software for tracking organisms," *Methods in Ecology and Evolution*, vol. 9, pp. 460-464, 2018.

[32] R. B. Sherley, T. Burghardt, P. J. Barham, N. Campbell and I. C. Cuthill, "Spotting the difference: towards fully-automated population monitoring of African penguins Spheniscus demersus," *Endangered Species Research*, vol. 11, pp. 101-111, 2010.

[33] J. P. Crall, C. V. Stewart, T. Y. Berger-Wolf, D. I. Rubenstein and S. R. Sundaresan, "Hotspotter_patterned species instance recognition," in *2013 IEEE workshop on*

applications of computer vision (WACV), 2013.

[34] N. Dhulekar and L. Bange, "Automatic, Non-Intrusive Identification of Jaguars In the Wild," 2011.

[35] C. Town, A. Marshall and N. Sethasathien, "Manta Matcher: automated photographic identification of manta rays using keypoint features," *Ecology and evolution,* vol. 3, pp. 1902-1914, 2013.

[36] L. Hiby, P. Lovell, N. Patil, N. S. Kumar, A. M. Gopalaswamy and K. U. Karanth, "A tiger cannot change its stripes: using a three-dimensional model to match images of living tigers and tiger skins," *Biology letters,* vol. 5, pp. 383-386, 2009.

[37] M. J. Kelly, "Computer-aided photograph matching in studies using individual identification: an example from Serengeti cheetahs," *Journal of Mammalogy,* vol. 82, pp. 440-449, 2001.

[38] M. Lahiri, C. Tantipathananandh, R. Warungu, D. I. Rubenstein and T. Y. Berger-Wolf, "Biometric animal databases from field photographs: identification of individual zebra in the wild," in *Proceedings of the 1st ACM international conference on multimedia retrieval*, 2011.

[39] A. Ardovini, L. Cinque and E. Sangineto, "Identifying elephant photos by multi-curve matching," *Pattern Recognition,* vol. 41, pp. 1867-1877, 2008.

[40] S. Li, J. Li, W. Lin and H. Tang, "Amur Tiger Re-identification in the Wild," *arXiv preprint arXiv:1906.05586,* 2019.

[41] M. Korschens, B. Barz and J. Denzler, "Towards Automatic Identification of Elephants in the Wild," *arXiv preprint arXiv:1812.04418,* 2018.

[42] D. Deb, S. Wiper, S. Gong, Y. Shi, C. Tymoszek, A. Fletcher and A. K. Jain, "Face recognition: Primates in the wild," in *2018 IEEE 9th International Conference on Biometrics Theory, Applications and Systems (BTAS)*, 2018.

[43] G. S. Cheema and S. Anand, "Automatic Detection and Recognition of Individuals in Patterned Species," *Joint European Conference on Machine Learning and Knowledge Discovery in Databases,* pp. 27-38, 2017.

[44] L. Bergamini, A. Porrello, A. C. Dondona, E. Del Negro, M. Mattioli, N. D'alterio and S. Calderara, "Multi-views Embedding for Cattle Re-identification," in *2018 14th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*, 2018.

[45] C.-A. Brust, T. Burghardt, M. Groenenberg, C. Kading, H. S. Kuhl, M. L. Manguette and J. Denzler, "Towards automated visual monitoring of individual gorillas in the wild," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[46] A. Freytag, E. Rodner, M. Simon, A. Loos, H. S. Kuhl and J. Denzler, "Chimpanzee faces in the wild: Log-euclidean cnns for predicting identities and attributes of primates," in *German Conference on Pattern Recognition*, 2016.

[47] X. Zhang, H. Luo, X. Fan, W. Xiang, Y. Sun, Q. Xiao, W. Jiang, C. Zhang and J. Sun, "Alignedreid: Surpassing human-level performance in person re-identification," *arXiv preprint arXiv:1711.08184,* 2017.

[48] L. Zheng, H. Zhang, S. Sun, M. Chandraker, Y. Yang and Q. Tian, "Person re-identification in the wild," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[49] G. Ciaparrone, F. L. Sánchez, S. Tabik, L. Troiano, R. Tagliaferri and F. Herrera, "Deep Learning in Video Multi-Object Tracking: A Survey," *Neurocomputing, Elsevier,* 2019.

[50] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, R. Pflugfelder and J.-K.

Kamarainen, "The Seventh Visual Object Tracking VOT2019 Challenge Results," in *The IEEE International Conference on Computer Vision (ICCV) Workshops*, 2019.

[51] D. Mykheievskyi, D. Borysenko and V. Porokhonskyy, "ODESA: Object Descriptor that is Smooth Appearance-wise for object tracking tasks," 16 June 2019. [Online]. Available: https://motchallenge.net/tracker/SRK_ODESA. [Accessed December 2019].

[52] Q. a. Z. L. Wang, L. Bertinetto, W. Hu and P. H. Torr, "Fast online object tracking and segmentation: A unifying approach," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[53] C. Xin, Bao and J. Tsotsos, "Fast Visual Object Tracking using Ellipse Fitting for Rotated Bounding Boxes," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2019.

[54] N. Wojke, A. Bewley and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *2017 IEEE International Conference on Image Processing (ICIP)*, 2017.

[55] Raspberry Pi, "Raspberry Pi 3 Model A+," [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-a-plus/. [Accessed December 2019].

[56] Raspberry Pi, "Camera Module," [Online]. Available: https://www.raspberrypi.org/documentation/hardware/camera/README.md. [Accessed December 2019].

[57] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

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

[59] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," *European conference on computer vision*, pp. 818--833, 2014.

[60] K. Simonyan, A. Vedaldi and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv preprint arXiv:1312.6034*, 2013.

[61] M. Körschens and J. Denzler, "ELPephants: A Fine-Grained Dataset for Elephant Re-Identification," in *The IEEE International Conference on Computer Vision (ICCV) Workshops*, 2019.