Visual Monitoring for Wildlife : Detection and Re-identification

A THESIS

submitted by

ANANYA

in partial fulfilment of the requirements
for the award of the degree of

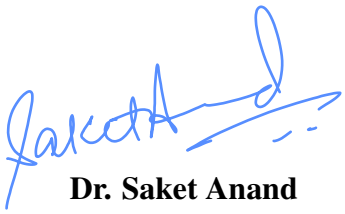
MASTER OF TECHNOLOGY

ELECTRONICS AND COMMUNICATION ENGINEERING
INDRAPRASTHA INSTITUTE OF INFORMATION TECHNOLOGY DELHI
NEW DELHI- 110020

May, 2023

THESIS CERTIFICATE

This is to certify that the thesis titled **Visual Monitoring for Wildlife : Detection and Re-identification**, submitted by **Ananya**, to the Indraprastha Insititute of Technology, Delhi, for the award of the degree of **Master of Technology**, is a bona fide record of the research work done by her under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.



Dr. Saket Anand
Thesis Supervisor
Associate Professor
Dept. of CSE and ECE
IIIT Delhi, 110020

Place: New Delhi

Date: May, 2023

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor, Dr. Saket Anand, for his invaluable guidance, support, and encouragement throughout my research journey. I will forever be grateful of him for giving me this opportunity, for believing in me and for his expertise, insights, and feedback.

I would also like to extend my heartfelt appreciation to Ashima Garg, for her generous assistance, collaboration, and helpful discussions. I have learned so much from her, both personally and professionally. I am grateful for her mentorship, support and contribution specially in the animal detection work.

I am truly fortunate to have had the opportunity to work with such amazing individuals at IIITD, and I will always cherish the knowledge and experience gained during this journey.

I would also like to thank Dr. Shikha Bisht, Dr. Ayan Sadhu , Prof. Y.V. Jhala and Prof. Qamar Qureshi from the Wildlife Institute of India (WII) for their feedback on the segregation application (CaTRAT) and using it for All India Tiger Estimation (AITE), 2022.

This work was supported partly by SERB, Govt. of India, under grant no. CRG/2020/006049 and partly by the Infosys Center for Artificial Intelligence at IIIT-Delhi.

Lastly, I would like to thank my family and friends for constantly supporting and motivating me. Your encouragement and belief in me have been invaluable, and I am truly grateful for your presence in my life.

ABSTRACT

Visual wildlife monitoring of animals requires detection for species-level categorization and re-identification (Re-ID) for population estimation of an individual species. Traditionally, the monitoring is done via GPS collars which are invasive, but the advent of camera traps has given a convenient, non-invasive and inexpensive alternate method for monitoring of wild animals. This camera-trap image data can be used with AI-based algorithms for detecting animal presence, species-level categorization, as well as individual identification or animal biometrics for certain species. To this end, we have developed the Deep Learning (DL) based species categorization module for the Camera Trap Data Repository and Analysis Tool (CaTRAT), which was used by the Wildlife Institute of India (WII) for processing the camera trap images during the All India Tiger Estimation 2022.

Beyond species-level segregation, deep learning approaches have also shown good performance for re-identification tasks. However, these methods often fall short, when encountered with fine grained patterned species like tigers and leopards, both in terms of performance as well as interpretability. This limits their usability by conservation officials and practitioners. In this work, we propose an end-to-end network to learn feature representations, keypoints, and their descriptors. The keypoints enable the model to: a) learn better discriminative feature representations and b) focus on salient regions (patterns) of the image. It is important to note that while training, we don't have ground-truth keypoint and descriptor annotations but only the label information. A pre-trained, DenseNet model is fine-tuned by a classification cross-entropy loss regularized by a pairwise Jensen-Shannon divergence. Further, feature map normalization regularizes the descriptor loss. The fusion of the keypoint attention feature map in the network helps focus on regions important for animal biometrics. We then evaluate the efficacy of our model on two datasets of patterned species, namely Amur Tigers and Leopards, under different biometric evaluation protocols: mAP, top-1, top-5, closed-set identification, open-set identification, and verification.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

WII	Wildlife Institute of India
AITE	All India Tiger Estimation
CaTRAT	Camera Trap Data Repository and Analysis Tool
YOLO	You Only Look Once
IDs	Identities
RE-ID	Re-identification
ML	Machine Learning
DL	Deep Learning
CNN	Convolutional Neural Network
SIFT	Scale Invariant Feature Transform
SFM	Structure From Motion
ATRW	Amur Tiger Re-identification in the Wild
JS	Jensen-Shannon
KL	Kullback-Leibler
AI	Artificial Intelligence

CHAPTER 1

Introduction

Wildlife is an integral part of our ecosystem and helps to maintain biodiversity. The wildlife population serves as an indicator of environmental health, especially the predator species like tigers and leopards. But in the past, it declined due to habitat loss, poaching, and human-wildlife conflict, which led them to become endangered. As per the Tiger Status report [47], the tiger population in India dropped to an alarming low (1411) in 2006. This motivated conservation efforts by authorities and consistent monitoring to assess the impact of policies around their conservation. As per the All India Tiger Estimation (AITE), in 2022, the minimum tiger population is 3167, a considerable increase from 2006. Since consistent population monitoring plays a vital role in their conservation, we need biometrics for wildlife. Visual patterns on an animal's body are a reliable identifier for their biometrics as an alternative to traditional pugmark tracking. These patterns can be captured via digital cameras, and the images could be used to identify individuals.

With the advent of technology, digital camera traps have become a popular way to collect data as they are readily available and affordable. They are either heat or motion-triggered devices which can capture both photos, for population estimation and videos, for behavioral studies of specific animals. They serve as suitable alternate data collection devices requiring minimal human intervention. Unlike GPS collars, they are non-invasive and a cheaper alternative. But due to their large scale deployment, they generate huge amounts of data, making it difficult to process and manage manually. Automating this process of identifying species and individual animals can reduce manual labor costs and speed up the data processing. It can decrease the time to draw insights from the raw camera trap data, reducing the policy-turn-around time.

Figure 1.1 shows the process of population density estimation of animals used in the Phase III of All India Tiger Estimation (AITE) as per the Status of Tigers report, 2022 [47]. Firstly, two cameras are installed opposite to each other in every 2 km^2 cell of the 100 km^2 grid to capture the animal's left and right sides. Extensive sign surveys identify the location of the camera trap in each cell to maximize animal capture. Then,

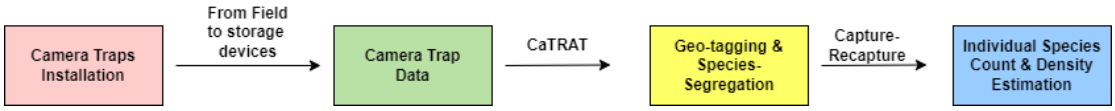


Figure 1.1: Animal population density estimation process

the data is retrieved from the field storage devices for analysis. As per the report [47], camera traps were placed at 32,588 locations spread across 174 sites leading to massive image data. These images are now geo-tagged to integrate spatial information and then segregated into respective species folders with the help of Camera Trap Data Repository and Analysis Tool (CaTRAT), an auto-segregation tool discussed later. This segregated data is now used for individual species population estimation. The traditional capture-mark-recapture method is used for individual recognition and density estimation for patterned species like tigers and leopards. Here, unique patterns on an animal's body are used to identify whether an image belongs to a previously seen individual.

We have identified two critical sub-tasks that can be automated in the visual animal biometrics process for the animal population density estimation. They will be discussed in further sections, namely: Animal detection and Re-identification.

1.1 Animal Detection

Detection is a joint task of identifying/ classifying and localizing multiple objects in an image. Thus, for an image, the task is to create a bounding box around the object and assign a class to it. It is usually formulated as a multi-task learning problem to optimize the model for both sub-tasks during the training phase. It is a crucial task in computer vision and finds application in video surveillance, crowd counting, anomaly detection, self-driving cars among them. It is excellent for tracking the movements of objects in images/videos.

It also finds application in detecting animals in the images captured by the camera traps. These images are captured by sensing the motion or body heat of the animals and range in crores after the final collection. Around four crores seventy lakhs camera trap pictures were processed in the AITE 2022 [47]. Many images that need to be processed belong to different categories, including persons, vehicles, blanks, and wild animals. Thus, species-level segregation becomes necessary before individual animal

identification for their population estimation. Furthermore, it has been observed that nearly 70 % of these images are blank or empty due to false triggers (plant/leaf movements, forest fires, or winds). Thus, it is futile to sift through them manually instead of auto-segregation. Considering all these factors, we need automated animal detection techniques, and the object detectors like YOLO [36], faster-RCNN [38] etc. come in handy. It is important to note that we perform detection instead of classification because there can be more than one animal in an image. It is also helpful in distance sampling methods famous for population estimation of non-patterned species where the detection of instances of animals in a region is needed. Moreover, generating a bounding box around the detected animal on a test image enhances the model's explainability, instilling confidence even in a non-expert.

Popular object detection models like YOLO [36] or Faster-RCNN [38] require both ground-truth labels and bounding box information. But we only have one label per image as ground-truth for nearly training 1,27,000 images. The bounding box annotations of training data are not given. Furthermore, it is cumbersome to generate these annotations manually considering the size of the dataset. Thus, we use an existing animal detection model, MegaDetector [2], which exhibits excellent performance in detecting animals, persons, and vehicles for the same. However, it does not perform species-level identification.
In our work, we train a network for species detection using a two-stage training framework :

1. Generating bounding-box annotations using MegaDetector [2]
2. Training YOLO-based [36] object detector on custom dataset

This pipeline will be discussed in detail in further chapters. Our model performs reasonably on the test dataset with an overall accuracy of 95%. In addition, we developed a tool, CaTRAT, that utilizes the model's predictions to categorize images into separate class-specific folders automatically. Using the tool doesn't require any machine learning background, making the Artificial Intelligence(AI) technology accessible to all.

1.2 Animal Re-Identification

In traditional capture-mark-recapture strategy [47] for minimum population estimation of endangered species like tigers and leopards by WII, the camera traps images are used to identify unique tigers or leopards by manually sifting for unique flank or rosette patterns. To remove the duplicates using stripe matching, they use an interactive software, Extract-Compare [17] which requires manually placing seeds/points(15-20) on prominent regions of each image to fit a 3D model and extracting the pattern. To reduce manual efforts, we shift focus on the automatic identification of unique tigers/leopards that can be achieved with the help of Re-Identification (Re-ID). Re-ID refers to matching images in the gallery set to a given query image to assign it an ID. This technique can be used to identify if a new image is of a previously seen animal or actually a new one. Although Re-ID can be applied to subjects such as people, vehicles, or animals, the focus has traditionally been on human re-identification [50], [34], [43]. However, this has shifted towards re-identifying animals and vehicles in recent years. In this work, we focus specifically on the Re-ID of endangered patterned species like tigers and leopards.

Monitoring the population of the endangered species can help in their conservation and assess the impact of policies around it. As discussed earlier, the data from the camera traps can be useful for individual animal identification. But the processing of this vast amount of data requires a lot of effort and resources if sifted through manually. Modern computer vision and deep learning methods have emerged as a bridge for the same.

After deep-learning frameworks showed remarkable performance in the ImageNet [7], [40] challenge, transfer learning has become a de-facto for fine-grained classification tasks as it reduces the training time significantly. Further, it has become a widely accepted technique for animal re-identification [46], [6], [49] too. In this work, we also use a pre-trained model and fine-tune it by minimizing a cumulative loss function, which is discussed in the later sections. The task is challenging due to a variety of issues such as poor image quality, lighting fluctuations, pose variations, and a limited number of images per identity in the training set.

Some of the available animal Re-ID works [26] try to overcome the pose variation challenge by using animal-pose information for predictions while others [25] require

part-pose annotations like limbs and trunks. This extra information is not universally available for datasets. Another work by [49] considers local and global features by vertical and global average pooling during training. For computing the loss for the local features, the image is divided into vertical parts, which are compared with their corresponding part in the other image. This comparison is intuitive but considers the whole vertical block of the image, comprising critical regions like the head or tail of the species combined with a lot of background information. In work by Shukla et al. [46] for tiger re-identification, training is single-step, but the inference is a multi-step process that includes distance ranking, left/right flank reordering, and SIFT re-ranking. SIFT is used to rearrange the predictions based on keypoint matching and helps in performance enhancement but only at inference. In our work, we use the keypoint information during training as well. The critical challenge in keypoint detection is the need for ground-truth keypoint annotations in the datasets. Further, keypoint matching involves identifying corresponding and confounding points, which again becomes challenging without ground-truth values.

Our idea of incorporating keypoints in the re-identification pipeline is inspired by how humans identify the patterned species. The stripe patterns play an essential role in recognizing and differentiating between tigers, as demonstrated by Hiby et al. [17]. Similarly, rosette patterns on the leopard's body are crucial for their identification [3]. Our proposed approach enable the network to focus on these critical regions by learning global and local features. The global features generalize the entire object by describing the whole image, whereas the local features represent the crucial points of an object in the image.

1.3 Contributions of this Thesis

Our contributions to animal detection and re-identification through this work include the following:

1. Species-level categorization and segregation of images for 98 species with an overall accuracy of 95.18%.
2. An end-to-end network trained to learn feature representation and keypoint detection. Joint learning helps the network learn better discriminative features and focus on the relevant regions for predictions.

3. Keypoint detection through the proposed network without the need for extra annotations or ground-truth keypoints and descriptors.

4. Using Jensen-Shannon(JS) Divergence instead of Kullback-Leibler(KL) Divergence along with keypoint attention feature map infused network to improve performance.

5. An auto-segregation tool, CaTRAT, which was used by WII for AITE 2022 [47].

CHAPTER 2

Literature Review

This chapter presents a comprehensive review of existing techniques for object detection, re-identification, and keypoint detection and matching. By examining the current state of the art for both detection and re-identification, we aim to identify the gaps and challenges that can be addressed in our methodology. We also discuss DL-based keypoint detection and matching techniques with the aim to integrate the same in our proposed network.

2.1 Object Detection

Detection is a prevalent paradigm in computer vision. The classical vision approaches in object detection include sliding window approaches like DPM [12], and PASCAL [11], where a classifier is applied on a dense image grid. These approaches use three stages, namely: framing candidate regions (using sliding window), extracting features (using SIFT [28], HOG [5]) for detection, but soon, they got replaced by neural network based approaches in the modern times.

In the deep learning paradigm, object detection methods can be categorized into two broader categories: one-stage detectors like YOLO [36], SSD [27], Retina-Net [24] or two-stage detectors like R-CNN [15], Faster R-CNN [37],FPN [23], Mask-RCN [16]. The first stage in a two-stage detector includes extracting sparse region proposals that could contain all the objects in an image, and the second stage processes them for classification. This approach was first used in R-CNN [15] and was improved over the years. Faster R-CNN combined Region Proposal Network (RPN) [37] and Fast R-CNN [14] integrated into a single end-to-end network that improves speed along with accuracy. But these methods are not generally preferred for real-time applications due to slower speed. The one-stage detectors are faster and more straightforward as they skip the region proposal stage and detect objects in a single network. There is a speed/accuracy trade-off. Hence their performance suffers a bit. The one-stage approach was proposed

in Overfeat [44] for the first time. Now-a-days, SSD [27], YOLO [36], RetinaNet [24], CornerNet [21] are the popular approaches. SSD, a single-shot detector, covers objects using multiple feature maps at different scales called anchor boxes and then directly classifies and refines each anchor box. Although faster, the average precision drop 10-20% as compared to two-stage detectors. In Yolo [36], an end-to-end network generates bounding boxes around objects and predicts class probabilities in images in one evaluation. This unified approach makes it fast, efficient, and highly precise, making it a perfect choice for use in the fields. It was pre-trained on the Coco dataset, a large-scale object detection, segmentation, key-point detection, and captioning dataset. The image is divided into regions, and the algorithm predicts probabilities and bounding boxes for each region. The Ultralytics team maintains this algorithm, and has further been modified and improved in its successive versions. We use Yolo version-5 in our work.

For generating bounding box annotation for our training data, we used an animal detector by Microsoft AI for Earth team, Megadetector [2], which detects animals, vehicles, and people in a given image. This detector performs remarkably in detecting animals (but not identifying), even on species not seen during training, in diverse ecosystems worldwide, making it ideal for our use case. Megadetector v5.0 was modified to detect vehicles, people, and animals at the finest level of the hierarchy for the train and validation sets, which would then act as our true labels.

2.2 Re-Identification

With the advancement of computer vision techniques, the performance for tasks like object detection and localization [13] has greatly improved. Further, using CNN for object classification [20],[9] showed great success over classical computer vision and machine-learning-techniques, which paved the way for deep-learning-based computer vision [36], [42], [27]. These methods have performed well on various tasks like animal detection and classification. Further, deep learning networks are also widely used to assist ecological studies [35]. After the ImageNet[40] challenge, transfer learning gained popularity. It involves fine-tuning the networks that are pre-trained on large scale dataset for fine-grained object classification tasks[4],[1], [48]. In our work, we also utilise a pre-trained deep network, densenet121 [18], fine-tuned on our dataset for

the re-identification task.

The classical ML models use handcrafted features to predict a particular image's class. Still, DL-based models learn these features in analogy to how humans differentiate or identify two images. Many deep learning methods outperformed the classical machine learning methods for the task of animal re-identification which will be discussed further.

This paper [46] proposes a DL-based approach combined with SIFT to rank gallery images against a query image. The model is Densnet-based, fine-tuned by minimizing cross-entropy loss and regularised by pairwise KL divergence loss. We use this model/network as our base architecture in our approach. During testing, it involves first finding class scores based on the DL model, using them to find cosine similarity, then use left/right flank separation for reordering, and finally using SIFT for re-ranking.

Another paper [45] proposes a PFID(primate face identification) system based on DL that uses guided pairwise KL divergence loss along with cross-entropy loss. The proposed model is compared with standard baselines(ResNet, Densenet, etc.). Evaluation protocols are standard bio-metric assessment protocols done under different settings: Classification, closed-set & open-set identification, and verification. These protocols are generally used to analyze the performance of face recognition systems. Another work [6], a new CNN architecture, PrimeNet, for wildlife identification on small datasets, shown comparison with state-of-art approaches, and evaluation is done on verification, open-set, closed-set. We also use these protocols in our experiments.

Another methodology, as presented in [26] a multi-camera pig tracking system for global identification using the YOLOv4 algorithm (deep learning-based detection algorithm) for detecting pigs in each camera view. It assigns local identity in that view using DeepSORT and then matches the identities from different camera views using homography to obtain global identity.

A different approach mentioned in [32] built a cattle identity classifier from videos to present the user with a set of images (e.g., 4) that contains the correct individual with a probability of more than 75 %. It uses self-supervised individual identifiers, contrastive learning, and gaussian mixture model cluster in the latent embedding to generate top-16 candidate labels. We also experimented with contrastive loss in our

work.

The animal Re-ID works in [25],[26] use animal-pose information for predictions. It requires part-pose annotations like limbs and trunks, which are not readily available for all datasets. Another work by [49] considers local and global features by vertical and global average pooling during training. For computing the loss for local feature, the image is divided into different vertical parts, and compared each corresponding part with the other image. Although this comparison is intuitive but considers the whole vertical block of the image, which comprises a lot of background along with important regions like the head or tail of the species.

Another work presents an approach [33] where re-identification of the Giant Sunfish is done using key-points matching and then selecting an identity based on the maximum matching score. The key-point detection is done using SIFT, RootSIFT, and SuperPoint (deep learning-based key-point detection). Then, key-point matching is done by finding features most similar with respect to distance using the Brute Force(inbuilt) method, which compares all elements in two distributions. It is guaranteed to find the best match. Then, ambiguous keypoints are dismissed by cleaning the matches based on distance to the nearest neighbor and distance to second the nearest neighbor; the threshold then cleans those matches.

In a different methodology [29], an effective model to learn pose-invariant image embedding using key points is proposed, which led us to explore more key-point-based techniques. They aligned the image embedding with a predefined order of the key points. They have proposed the KAE-Net model, a conceptually simple but effective model for learning pose invariant image embedding. This model generates key-point embedding and key-point heat map, which are compared with other samples to find loss that is backpropagated.

Thus, realizing the importance of keypoints-based techniques in re-identification, we will explore existing key-point detection methodologies in the further section.

2.3 Keypoint Detection and Matching

Traditional interest keypoint detection techniques include SIFT [28], FAST [38], and ORB [39]. SIFT [28] uses handcrafted features while FAST [38] is a machine learning algorithm using decision trees. The paper [30] is a comparative study of handcrafted image feature detectors and descriptors. However, recent approaches have explored deep learning algorithms to detect interest points.

Supervised approaches for keypoint detection are computationally expensive. Recent research interests lie in developing a self-supervised approach to detect keypoints. DeTone et al. [8] propose Superpoint, a CNN-based architecture for interest point detection and description/matching. It uses a pre-trained interest point detector trained on synthetic data. Superpoint follows detect and then describe approach. D2-Net [10], addressed the following joint detection and description approach. A single network plays the dual role of feature descriptor and detector. The model can be trained using pixel correspondences extracted from large-scale SFM reconstruction without further annotations. But it requires a depth map and camera pose information for detection and matching. Ono et al. [31] proposed LFNet, where a deep detector network returns scale-invariant keypoints while the descriptor network generates a descriptor for a patch around the keypoint detected. Superglue [41] uses an attention graph neural network and optimal matching layer for learning feature matching by finding partial assignments P between two sets (M, N) of local features (keypoint and visual descriptors). These features are taken from the deep front end like Superpoint [8]. Key.Net, as proposed by Axel et al. [19], combines both handcrafted and learned features for keypoint detection. Its architecture combines gradient-based feature extraction (Harris and Hessian detectors), learned features (CNN), and multi-scale pyramid representation. The feature maps from all scales are up-sampled, concatenated, and passed through 5x5 CNN to generate the final response map. Now, the multi-scale index proposal (M-SIP) layer splits response maps into grids multiple times (different windows) to compute the candidate keypoint position for each window.

Another work by Lin et al. [22] introduced a deep-network RK-NET which allows representation learning and keypoint detection through a single network. They introduced a Unit Subtraction Attention Module(USAM) between the original blocks of another deep network, ResNet-50, to extract the keypoint feature map. The input and

output feature maps have the same dimension, making them universal. In our work, we use the output of the USAM block as described in this paper to extract keypoints on the images.

CHAPTER 3

Animal Detection & Species Segregation

The animal population density estimation involves organizing the data into separate species folders before individual species counting. So, we train a custom model for animal detection on the dataset provided by the Wildlife Institute of India (WII). We will discuss details regarding the dataset and its challenges, the training pipeline of an animal detector, and integrating it to build an auto-segregation tool. We will also look at the performance of our model on different metrics and different misclassification scenarios.

3.1 Dataset Description & Challenges

After seeing an alarming drop in the number of tigers in India in 2006, Tiger Cell was established to conserve tigers by WII and it is still active. This cell conducts regular assessments of tigers, their co-predators, prey, and habitat across the country. They have installed digital camera traps at sites having the highest chance of photo-capturing tigers, leopards and other species. These are battery-operated cuddle-back C1 cameras with memory cards to save the captures when triggered by motion or heat. For All India Tiger Estimation (AITE), these cameras are set to photo capture mode instead of video. The data is retrieved from the memory cards weekly or bi-weekly. This data is unsegregated and requires a first round of species-level segregation before individual identification for population counting. The dataset shared by WII for training the animal detection model comprised nearly 1,80,000 images and 106 unique species. Figure 3.1 and 3.2 are samples from train set showing tiger, leopard and other wild animals. This dataset has many challenges including:

1. **Blurred images:** The images are sometimes blurred/unclear due to animal motion or flash, as shown in the first image of figure 3.1.
2. **Poor lighting:** Sometimes, the images have terrible lighting such that the animal is not visible, which can confuse the model, as seen in the second image of figure 3.2.

Figure 3.1: Sample images of tigers from the train set

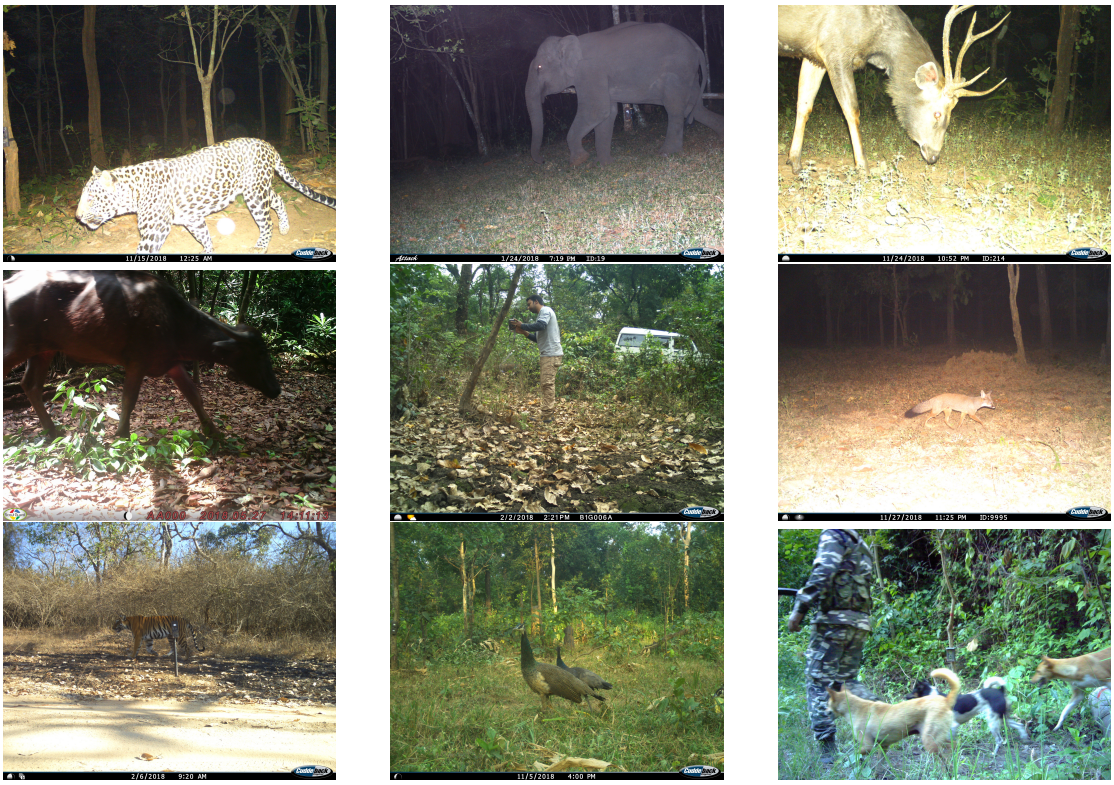


Figure 3.2: Other samples images from the train set

Figure 3.3: Morphologically(Mongoose) similar but different category samples from the train set

3. **Imbalanced dataset:** Figure 3.4 clearly shows that the dataset is class imbalanced. Also, figure 3.5 shows the number of images per class varies from 6-4715, and most classes have less than 1500 images. Nearly eight species have less than ten images; we ignore them, leaving 98 classes. Furthermore, 16 species out of these have less than 100 images.

4. **Limited annotations:** Initially, only one label per image is available even for images with multiple species. Further, ground-truth bounding box information is not available.

5. **Animal camouflage and occlusions:** Some images have animals camouflaged or occluded by objects that are difficult to recognize even by humans.

6. **Morphologically similar species:** Morphologically similar species are challenging to differentiate, like jungle and desert cats, different types of mongooses, etc. Figure 3.3 shows images from morphologically similar species, but each belongs to another category.

Figure 3.6 shows classwise histogram of image count per class for the validation set, which is 15% of the total data. From the histograms, we can see that the dataset is skewed i.e. a few classes have more number of images per class.

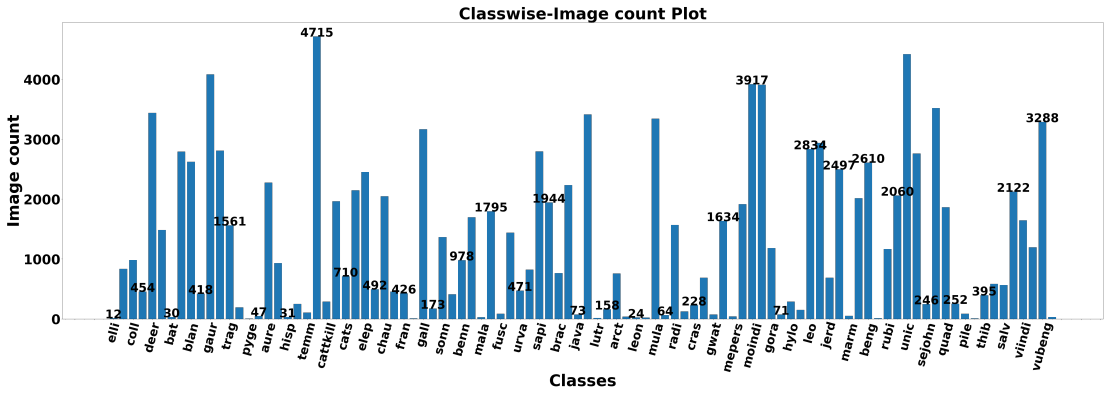


Figure 3.4: Histogram of image count per class of the train set

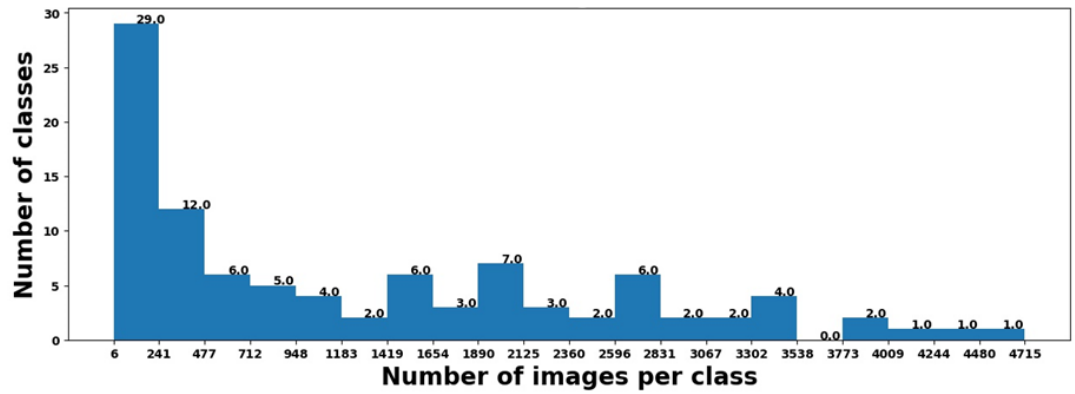


Figure 3.5: Histogram of number of classes having a particular number of images

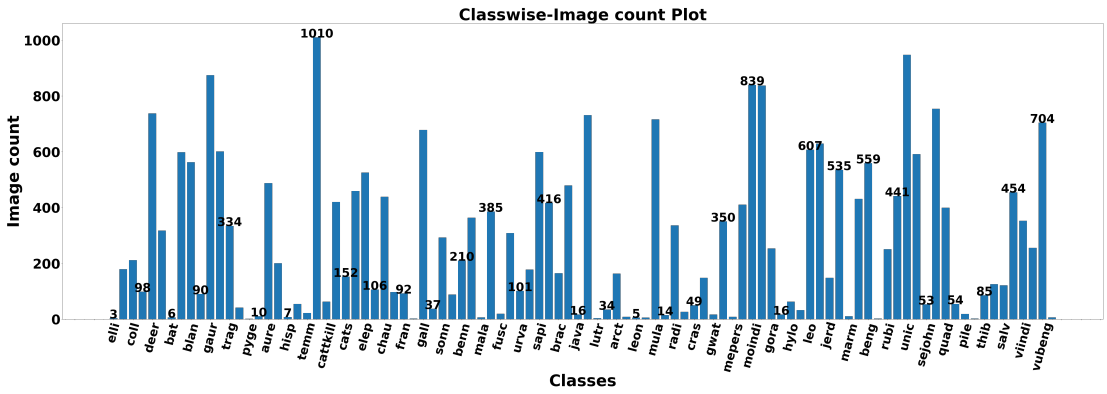


Figure 3.6: Histogram of image count per class of the validation set

3.2 Training Pipeline

As shown in figure 3.7, the training of animal detector is a two-stage framework which involves a MegaDetector model [2] and Yolov5 [36].

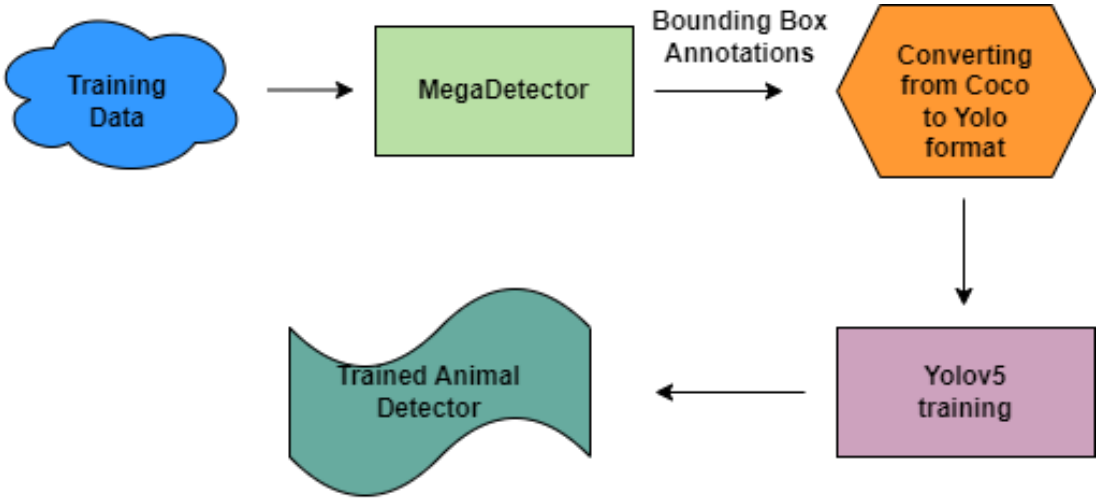


Figure 3.7: Species Detection Pipeline

The steps that were followed for training our animal species detector model involve:

1. We split the dataset into 70:15:15 ratio of train-val-test sets. Since we don't have bounding box annotations, the training set is passed through MegaDetector to generate bounding box annotations of objects, namely animals, vehicles, or persons, with a confidence score.

2. The bounding box annotations are converted from coco to yolo format using:

$$x_{yolo} = \frac{(x_{coco} + w_{coco})}{w_{img}}; \quad y_{yolo} = \frac{(y_{coco} + h_{coco})}{h_{img}}; \quad w_{yolo} = \frac{w_{coco}}{w_{img}}; \quad h_{yolo} = \frac{h_{coco}}{h_{img}} \tag{3.1}$$

3. Then, a .txt file, with class ID and annotations in the Yolo format(x_{yolo},y_{yolo},w_{yolo}, h_{yolo}) is generated for each image in training set except the blank images class(no

object of interest in the image). Figure 3.8 compares the bounding box format of Coco and Yolo.

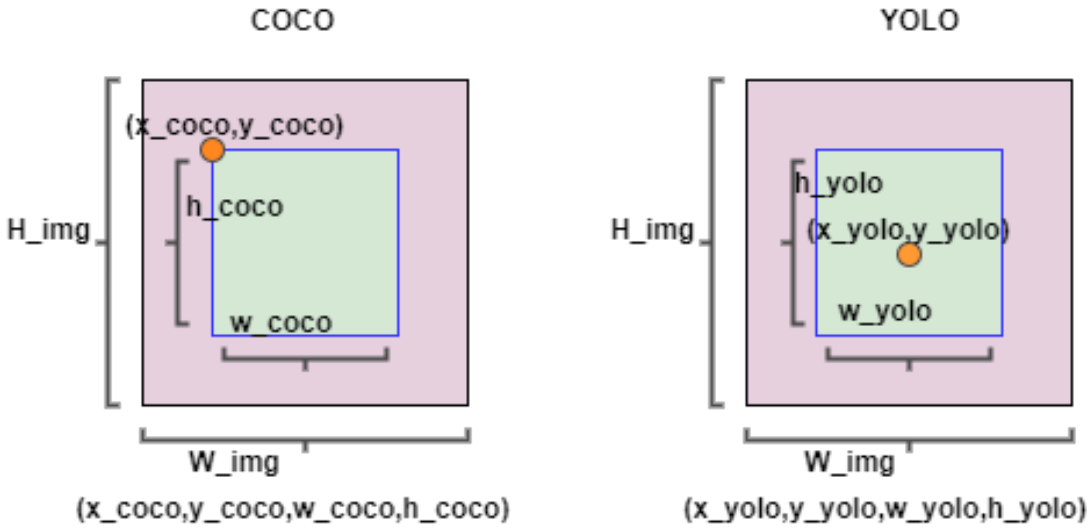


Figure 3.8: Bounding box format COCO vs. YOLO

4. Now, after merging all the images in one folder and the labels(.txt files) in another folder, yolov5 pre-trained model is fine-tuned on our custom dataset.
5. The trained model is then used to make predictions on the test dataset for evaluation.

Due to the limited annotations(one label per image), the first model resulted in misclassifications in multi-species images as discussed in the result section. We then took the below steps in training helped reduce model confusion to a great extent:

1. Using the bounding box annotations generated by MegaDetector, we created a list of images with more than one bounding box (with a confidence score of 0.8 or more).
2. WII helped manually annotate these images with the help of the LabelImg tool and generate corresponding .txt label files for each image.
3. We then perform the steps again for training the yolov5 model.

3.3 Implementation Details

We have used MegaDetector version 5.0 to generate bounding box annotations for the train and validation sets. Each annotation has a confidence score associated with it. We don't use images with low bounding box confidence scores (less than 0.1) for training

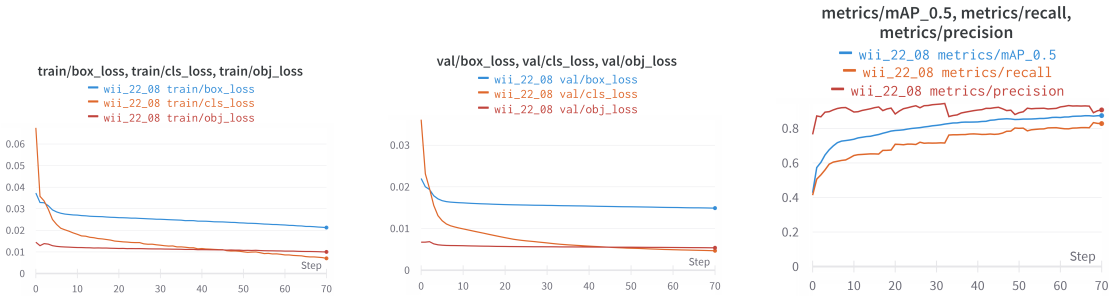


Figure 3.9: Train and validation loss plots along with mAP on the test set

and add them to the test set. The confidence score thresholds of 0.4 for animals and 0.5 for persons and vehicles are set for a bounding box annotation to be considered valid; else, it is rejected.

Using the annotations generated by the MegaDetector as ground truth, the Yolov5 (version 5) model is trained with the default hyperparameters on the training set for 70 epochs, image size 640, and batch size 16 with nearly 1,27,000 images. Figure 3.9 shows the loss plots for training and validation set. The validation set is used for performance evaluation of the trained model at every epoch and the model with the highest mAP value is selected as the best model. Figure 3.9 also shows mAP, precision and recall on the validation set.

3.4 CaTRAT-Auto Segregation tool

Using an AI-based model for species detection and segregation is not scalable to masses without or with very little technical expertise. The biologists analyzing the camera trap images don't have any background in the technology and hence need a user-friendly tool to perform segregation masking the intermediate steps.

CaTRAT(Camera Trap data Repository and Analysis Tool), an auto-segregation tool, closes this gap and provides a seamless way to segregate images into their respective species folder. The user needs to give the path of the source and destination folders, and the rest is taken care of. This tool comprises of :

1. A batch file to take source and destination folder as input from the user.
2. A trained model to generate labels and bounding box coordinates on source images.

3. Now, based on confidence score values, top-3 labels are selected. A global confidence threshold is set to 0.3 (1 being the highest). An image with all the predictions below the global threshold is assigned a 'blank' class. We also tried setting a local confidence threshold for each class instead of a global threshold. It was done with the help of precision-recall plots (figure 3.13) at different threshold values on the validation set. For species like tiger, leopard, and hyena, the threshold with a high recall and good precision was selected as we want very few false negatives. But for other species, a higher-precision threshold was chosen, and then classwise accuracy on the test set was calculated, as shown in figure 3.14.

4. These images are assigned the top-3 labels as tags in the order $'a\_species1'$, $'b\_species2'$, $'c\_species3'$ by modifying their metadata as shown in figure 3.12.

5. Now, these images are moved into their respective label/class folders, finishing the segregation.

Figure 3.10 shows auto-segregation process of images into their respective classes. Figure 3.11 shows a snapshot of our application as viewed by a user where the source

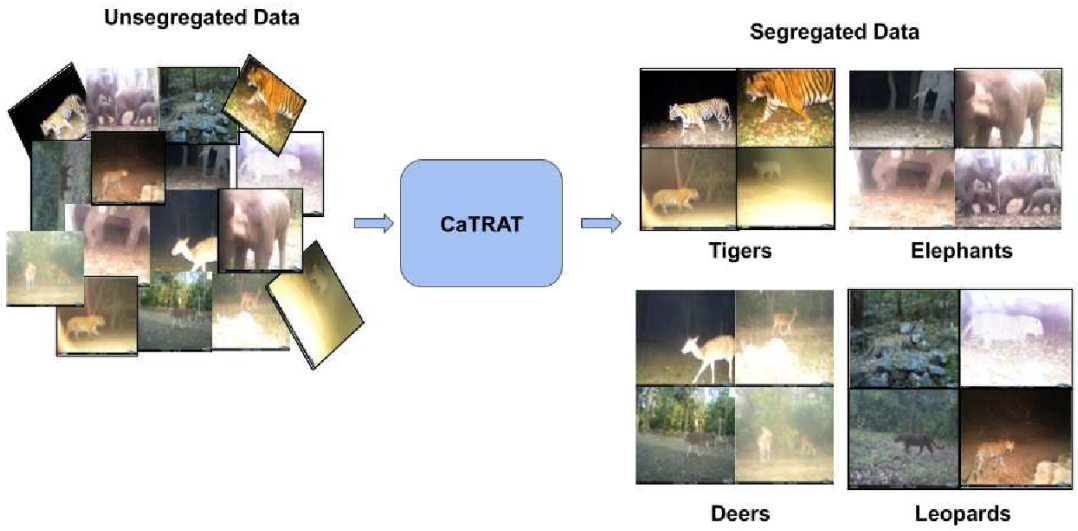


Figure 3.10: Auto-segregation process

and destination folders are given as an input. It also displays the current folder, the number of processed and remaining images and the processing stage (detection or segregation). Figure 3.12 shows the data segregation in the destination folder through sub-folders as different species. Further, all the images are tagged with their labels in the sub-folders, which helps to showcase multi-species images. Our tool has below features:

1. It auto installs all the required dependencies and displays the current stage logs to the user.

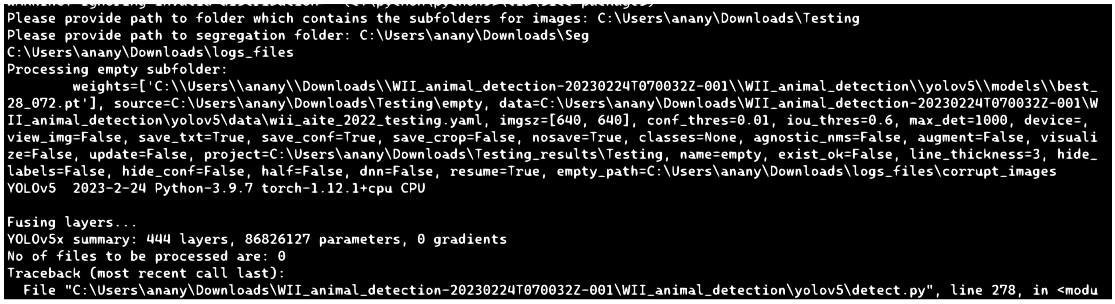
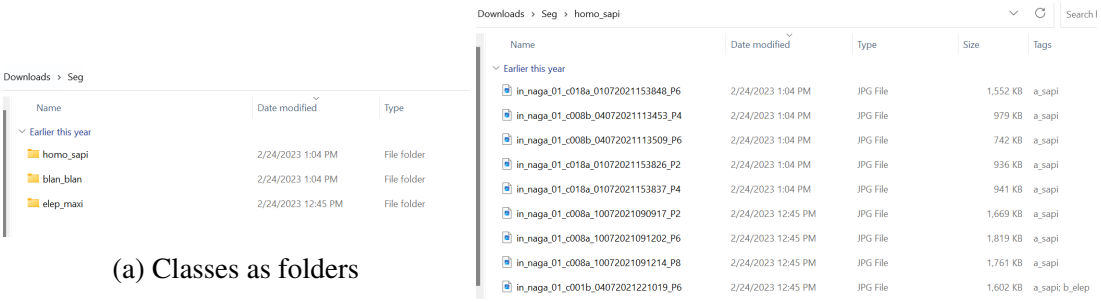


Figure 3.11: CaTRAT tool snapshot



(a) Classes as folders

(b) Tagged images for multi-class detection

Figure 3.12: CaTRAT tagging & segregation

2. It has a 'resume' feature in case the process stops at any stage (detection or segregation) since crores of images are processed in one go.

3. In case of multi-species detection, the image is tagged with top-3 detected species in order, which can help in identifying multi-species images or analysing model confusions among classes.

4. It creates visualization with the bounding boxes and confidence of each box on the images for quick glance on model's performance.

5. It can handle images with same names and don't skip them while segregation.

6. It also creates a folder named log_files which contains a list of processed_images, images containing multiple species, and a list of images with repeated names.

3.5 Results

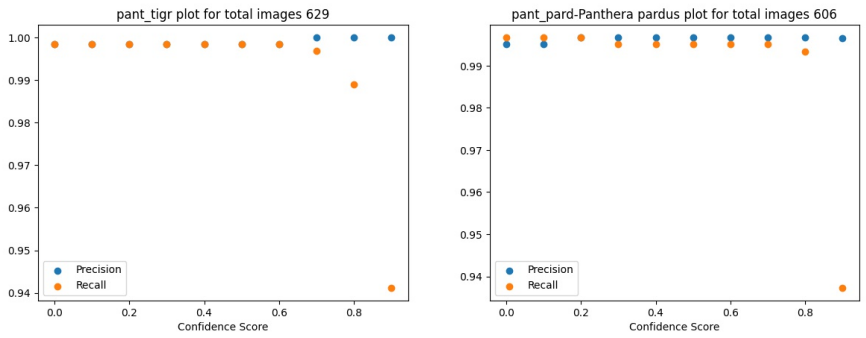
Table 3.1 shows the detection performance of our model for a few classes. Overall performance is better for classes with many training samples compared to those with fewer training samples. Also, the model's performance on endangered species like tigers, leopards, and hyenas is nearly 100% on the test set. Further, on the test set, we obtained an overall accuracy, precision, and recall of 98 %, 86.41%, and 84.38%, respectively.

Classwise Test Set Results						
Class-name	Training Samples	Test Samples	Misclassified	Accuracy	Precision	Recall
Tiger	2938	630	0	1	1	1
Leopards	2834	607	1	0.998	0.997	0.998
Striped Hyena	1944	417	2	0.995	0.998	0.995
Mouse Deer	2234	479	4	0.992	0.994	0.992
Brown Mongoose	3917	840	3	0.996	0.989	0.996
Macaque	85	18	7	0.611	0.688	0.611
Nilgiri Marten	123	26	6	0.769	1	0.769
Clouded Leopard	73	15	3	0.8	0.857	0.8
Overall Results	121946	26132	484	0.981	0.864	0.843

Table 3.1: Classwise accuracy, precision and recall on the test set

As mentioned earlier, we experimented with setting a variable threshold for each class based on the precision-recall trade-off for each threshold value on the validation set. A high recall value is required for some classes, like tigers, leopards, and hyenas since we want lower false negatives for them, while precision is prioritized for other classes.

Figure 3.13 shows the precision-recall plot for varying thresholds for two classes: Tiger and Leopard. We choose a threshold of 0.3 for both these classes based on these plots as it shows a high recall value and good precision on the validation set. Similarly, thresholding is done for other classes. Figure 3.14 shows the classwise accuracy of the



(a) Precision-recall plot for class 'Tiger'

(b) Precision-recall plot for class 'Leopard'

Figure 3.13: Exemplar Precision-recall plots

test set and figure 3.15 shows the number of images per class in the test set. We observe a reduction in the accuracy for classes with lower image count in the training and test set. Furthermore, accuracy for endangered species like tiger, leopard and hyena is close to 100 % on the test set.

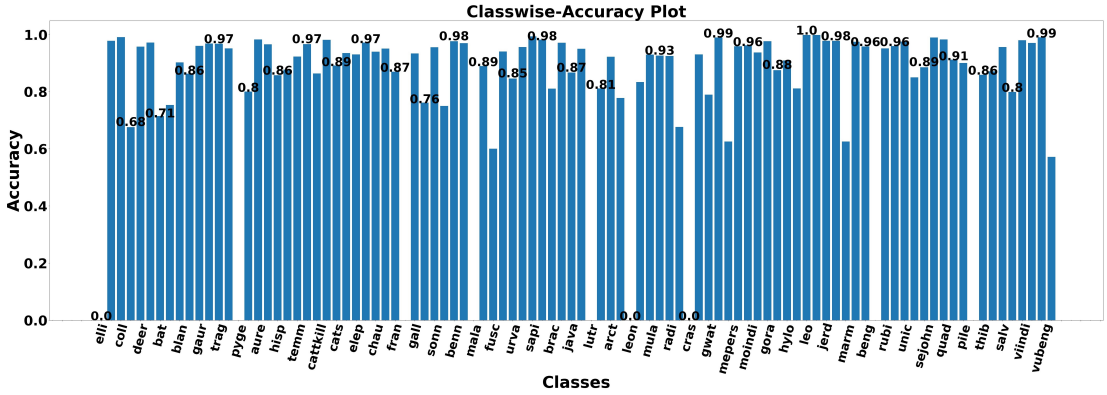


Figure 3.14: Classwise accuracy on the test set

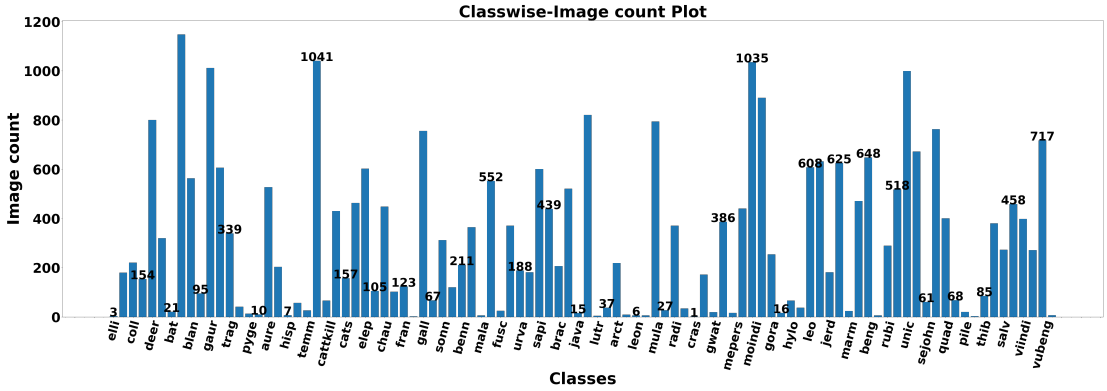
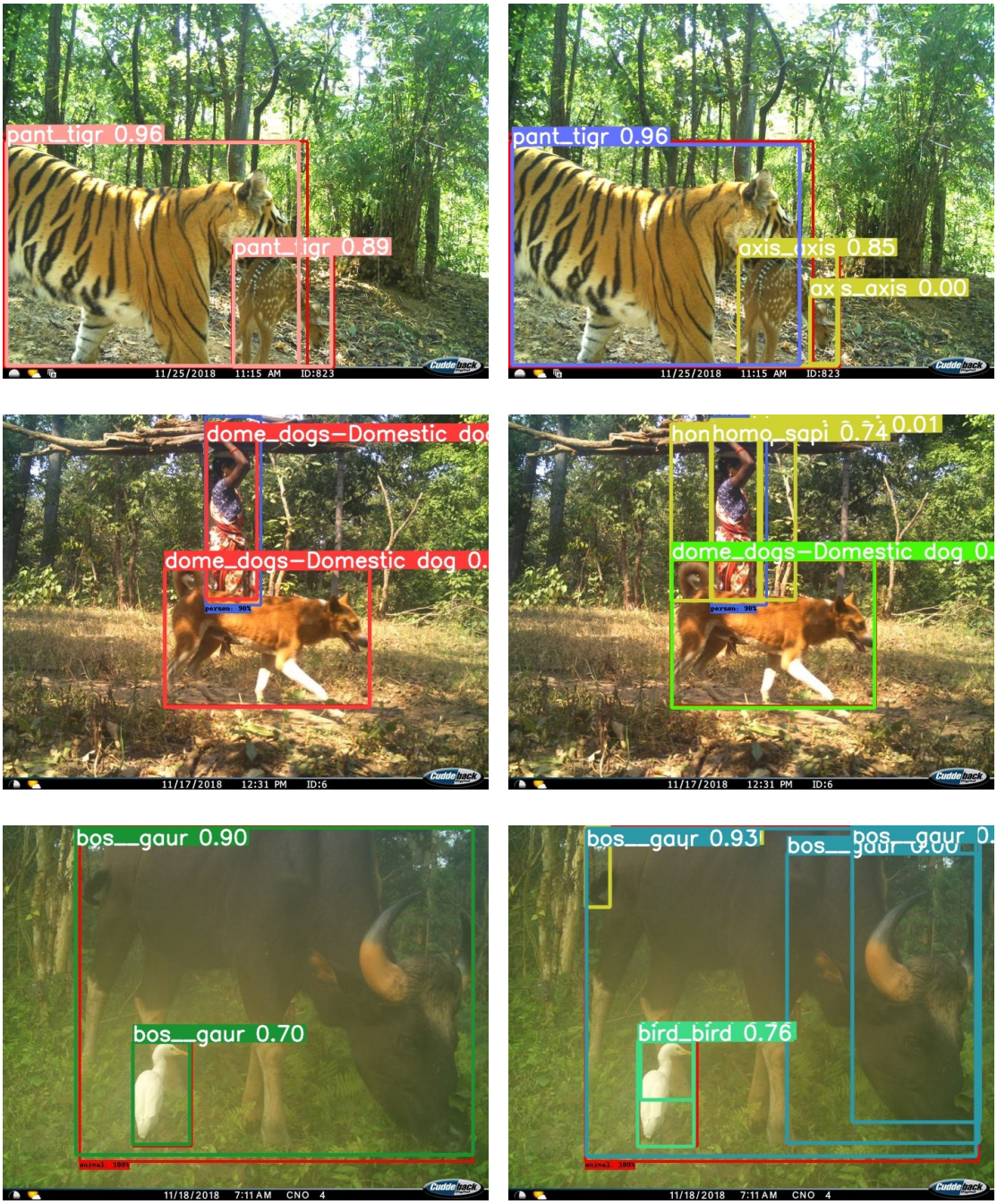


Figure 3.15: Histogram of image count per class of the test set

Figure 3.16 compares the result of two models, one trained with one label per image for all bounding boxes (even for multi-species images) and the other trained with multiple labels corresponding to each bounding box.We can see a clear improvement in the later model, which can detect multiple species like tiger and deer, human and dog, buffalo and bird in the images.

3.5.1 Misclassification scenarios

We have identified specific misclassified exemplars from the test set exhibiting a pattern corresponding to the dataset limitations. Figure 3.17 shows confusions due to bad lighting. Here, leopard is misclassified as bear, elephant as blank or no object and hyena as deer. Figure 3.18 shows misclassifcations due to occlusions like trees or animal camouflage. In figure 3.19, images are blurred and animal is not clearly visible causing confusion. Figure 3.20 displays confusion between morphologically similar species as they have very subtle differences between them.



(a) Model trained with one label per image for all bounding boxes

(b) Model trained with multiple label corresponding to each bounding box

Figure 3.16: Detection results on samples from test

(a) Leopard misclassified as bear (b) Elephant misclassified as blank (c) Hyena misclassified as Deer

Figure 3.17: Incorrect classification samples: Lighting fluctuations

(a) Cattle misclassified as blank (b) Bird misclassified as blank (c) Macaque misclassified as blank

Figure 3.18: Incorrect classification samples: Animal camouflage/occulsions

(a) Bird misclassified as blank (b) Blank misclassified as human (c) Hog deer misclassified as Barking deer

Figure 3.19: Incorrect classification samples: Blurring

(a) Hystrix indica species misclassified as Hystrix brachyura (b) Herpestes smithii misclassified as Herpestes edwardsii (c) Tetracerus quadricornis misclassified as Muntiacus muntjak

Figure 3.20: Incorrect classification samples: Morphologically similar species

CHAPTER 4

Animal Re-identification

After segregating the species into individual folders, we need to count unique animals in them for their population estimation. In this chapter, we focus on the re-identification of patterned species, tiger and leopard and propose a keypoint attention fused deep network for the same. The evaluation has been done on two datasets, namely: ATRW and Leopard ID 2022 described in the upcoming section.

4.1 Dataset Description

Table 4.1 describes the distribution (total number of samples and classes) of the Amur Tigers and Leopard dataset that we have used in our experiments. Amur Tiger Re-identification in the Wild(ATRW) dataset provides fixed train and test splits with 134 and 48 IDs, respectively. The train set is further split into train and validation in the 80:20 ratios based on their IDs.

For the Leopard dataset, we first remove the IDs with less than five samples leaving 193 IDs. Now, the test set is created with 33% IDs of the total dataset, and the remaining dataset is split into 80:20 ratios w.r.t IDs for the train and validation sets, respectively. Figures 4.1 and 4.2 show the train, validation, and test set distribution, i.e., the classes

Dataset	Total Samples / Total Individuals		
	Train	Val	Test
ATRW(Amur tigers)	1887/107	495/27	1269/48
Leopards	3171/103	852/26	2204/64

Table 4.1: Statistics of the datasets

or IDs vs. the number of samples in each for the Amur tiger and leopard datasets, respectively. For the ATRW dataset, the number of samples of the training set varies from 10-100, while it varies from 4-400 for the leopard dataset. We can see the class imbalance in both datasets from these figures, and the number of samples per ID is also quite low.

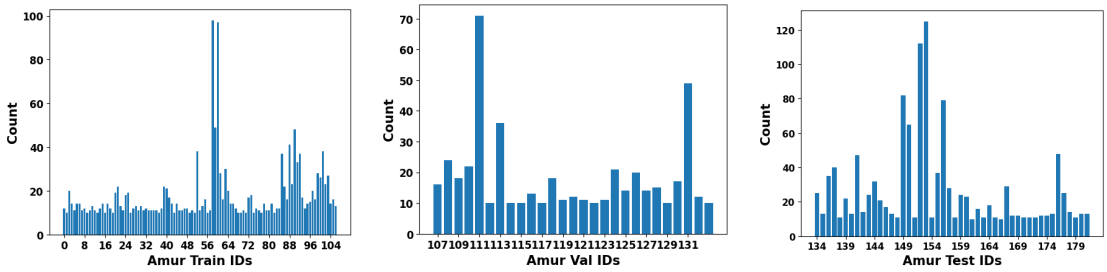


Figure 4.1: Train, validation and test set distribution of ATRW dataset

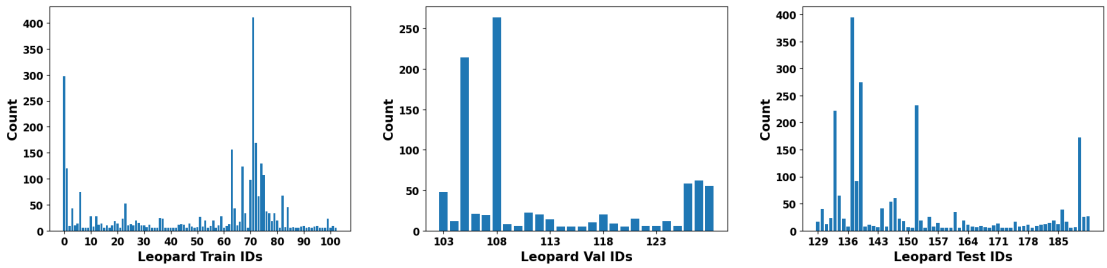


Figure 4.2: Train, validation and test set distribution of Leopard dataset

4.2 Dataset Challenges

Both ATRW and Leopards dataset are challenging since the images are captured in the wild and there are miniscule differences amongst them, difficult to be recognised even by humans. Some of the challenges are listed below:

1. **Poor image quality:** The images in the dataset suffer from blurring, occlusions and animal camouflage leading to poor image quality. Figure 4.3 shows leopard and tiger samples where identifying patterns are not clearly visible.

Figure 4.3: Samples with poor image quality

2. **Lighting fluctuations:** The dataset has both day and night captures of animals in the wild. Furthermore, extreme lighting diminishes the distinctive patterns that are essential for identification. Figure 4.4 shows some samples with poor lighting.

3. **Pose variations:** The animals in the dataset have variable poses, as shown in Figure 4.4, unlike face re-identification challenges. It can lead to misalignment between images making it difficult for the model to make correct predictions.

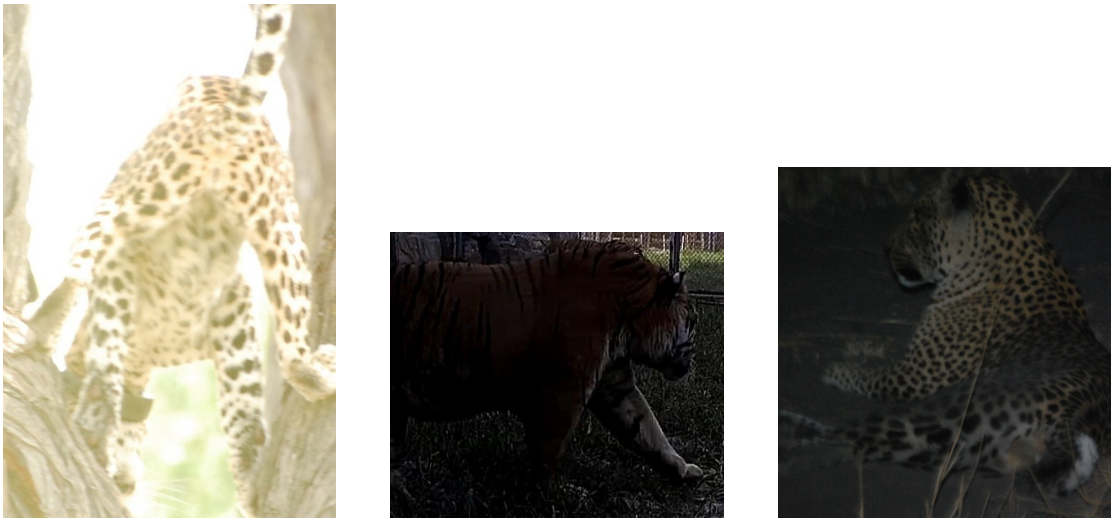


Figure 4.4: Samples with lighting fluctuation

4. **Fine-grained images:** The fine-grained datasets used in our task have very subtle differences between different IDs, making it difficult for the model to differentiate. Figure 4.6 shows samples from the ATRW dataset, with the first two images belonging to the same ID and the third to a different ID.

5. **Limited images per ID:** For the ATRW dataset, the number of images per ID varies from 10-100. For the leopard dataset, they range from 4-400. These are very few samples per ID for training the model compared to popular datasets with at least a thousand images per class.

4.3 Proposed Methodology

Figure 4.7 shows the architecture of the proposed training and evaluation methodology. The training images are applied with standard data transforms and used to train a modified deep network on a cumulative loss. The best model on close-set top-1 accuracy on validation set is selected for evaluation. The test set is divided into probe and gallery set. Each probe image act as a query and assigned an identity based on maximum cosine similarity with gallery set images. The details of each block in the architecture are :

1. **Training images:**The images for the Leopard dataset were cropped using the bounding box (around animal) information available before feeding them to the network. The ATRW dataset images were already cropped.

2. **Data transform:** We resize all the images to 224x224. Further, we used data augmentation techniques like random rotation(10 degrees), gray-scaling(0.2 probability), and color jitter (varying brightness, hue, contrast, and saturation) of the training images.

Figure 4.5: Samples with pose variation

(a) Query (b) Similar (c) Dissimilar

Figure 4.6: Samples showing fine-grained images from the ATRW dataset

3. **Homography:** Since we don't have ground-truth keypoint and descriptor matches, we use homography of original images for the same. We first generate a transform matrix which is a combination of projective, affine, and Euclidean transformations. The image is then warped using this matrix to create the image homography.

4. **Modified deep Network:** We have used a pre-trained deep network with USAM [22] block inserted after the first stage of the network. Another layer replaces the network's last layer with a number of neurons equivalent to the number of classes. Figure 4.8 shows a glimpse of the modified Densenet-121 network.

5. **Training loss:** The loss function is a cumulative sum of classification cross-entropy and pairwise JS-divergence loss, which acts as a regularizer, a descriptor loss, and a feature map regularizer. They will be discussed in detail in a further section.

6. **Probe & gallery set:** We have created probe and gallery sets for both close-set and open-set identification with at least 4 IDs in the gallery set per ID in the probe

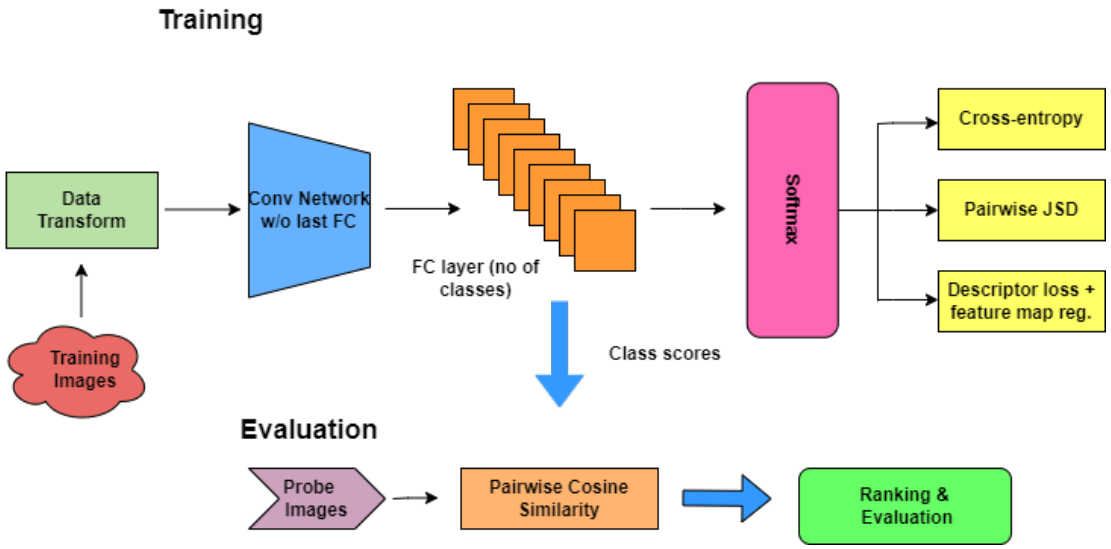


Figure 4.7: Training and Evaluation architecture for Re-identification

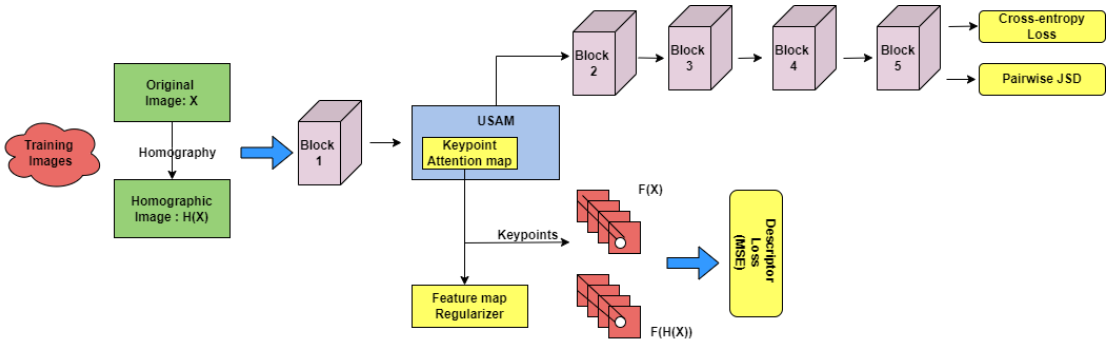


Figure 4.8: Modified Densenet-121 with USAM block after stage 1

set.

7. **Pairwise cosine similarity:** For an image in a probe set, we extract features from the second last layer of the network and find cosine similarity with all the images in the gallery set.

8. **Pairwise matching similarity:** It refers to the number of keypoint matches between probe and gallery set images.

9. **Ranking & Evaluation:** Based on the pairwise similarity, we assign an ID to the test/probe set image and then evaluate based on the metrics.

Figure 4.8 shows the convolution network fused with keypoint attention feature map. The keypoints are extracted from this feature map as described in section 4.3.1 and the locations are projected on the homography of original images. The descriptor loss uses these locations for corresponding matches between image and its homography. The USAM block is also described in detail in the section 4.3.1.

4.3.1 USAM and Keypoint extraction

USAM block as per [22] is inserted after stage 1 of the pre-trained network, and it comprises of below operation on the input feature map:

1. **Sum Pooling:** The sum pool operation is performed across the channel axis for feature map aggregation. This operation converts the feature map into a 2D tensor.
2. **Unit Subtraction Convolution:** Each element of the feature map is subtracted from all its neighbors in a 3x3 window, and then summed together. This operation is very close to convolution operation but instead of multiplication, subtraction is done. The kernel here has all the lements as one. This results in the output of the USC block. Further, applying batch-normalization and Relu activation on it, generates an attention feature map.
3. **Mask:** A binary mask of the same size as the attention feature map with the last two rows and columns as zero, is multiplied to it to avoid any keypoint detection at the boundaries.
4. **Fusion:** Now, this attention map is expanded to the original input feature map shape and added to the same to generate the output feature map of the USAM block.

Below are the steps to extract keypoints from the network:

1. Min-max normalization of the attention feature map.
2. Selecting top k keypoints in the order of highest values in the feature map and applying non-maximum suppression in a 12x12 window while selecting.
3. Resizing the feature map to the original image size and highlighting the keypoints.

4.4 Training Loss

For a classification setting, a multi-class cross-entropy loss(equation 4.1) is used to help the network learn the class or ID of an image by predicting the highest probability for the correct class.

$$L_c = -\sum_{n=1}^{N}\left(\sum_{c=1}^{C}\log\frac{\exp(x_{n,c})}{\sum_{i=1}^{C}\exp(x_{n,i})} * y_{n,c}\right) \tag{4.1}$$

where x is the input, y is the target, C is the number of classes, and N is the batch size. Along with cross-entropy loss, we use a pairwise JSD divergence loss(equation 4.2)

which helps the network use distribution over classes. It encourages similar distribution for the same IDs and dissimilar distribution for different IDs.

Let,

$$
\begin{aligned}
x &= \frac{p+q}{2} \\
L_s &= 0.5 * (KL(p,x) + KL(q,x)) \\
L_d &= 0.5 * ((0, m - KL(p,x))_+ + (0, m - KL(q,x))_+) \\
L_{JSD} &= yL_s + (1-y)L_d
\end{aligned}
\tag{4.2}
$$

where p and q are class probability vectors of two images, y is one if these images belong to the same class, else 0, m is the margin, $(\cdot)_+$ indicates max function, and $KL(\cdot,\cdot)$ is KL-divergence given by $\sum_{k=1}^{K} p_k \log \frac{p_k}{q_k}$

Further, we have employed a descriptor loss for transform invariant descriptors as per equation 4.3. The top k keypoints are extracted from normalized attention feature map (USC [22] at Stage 1). The descriptors vectors of the keypoints are extracted from the output feature map of the stage 1 USAM block at the keypoint location seen across the channel. For keypoint-based matching, a mean square loss is applied between top-k keypoint descriptors of the original image and their corresponding counterparts in the homography of that image. An average over the whole batch is backpropagated to the network.

$$
L_{desc} = \sum_{n=1}^{N} \left(\sum_{i=1}^{k} ||d_i - d_j|| \right) \tag{4.3}
$$

where *N* is the batch size, *k* is the total number of keypoints, *j* is the corresponding keypoint for location *i*, and d_i is the descriptor vector for keypoint *i*. The descriptor vector at a particular keypopint is a vector across channels at that location.

We also have an attention feature map normalization using the mean and standard deviation of the feature map norm(from the pre-trained model) of the training set which acts as a regularizer to avoid descriptor loss from driving the feature maps to null.

$$
L_{fm\_norm} = \frac{||X|| - \mu_{norm}}{\sigma_{norm}} \tag{4.4}
$$

where *X* is the Frobenius norm of the attention feature map, μ_{norm} and σ_{norm} are the mean and standard deviation of the Frobenius norm of the train set. Finally, the total loss is the cumulative sum of all the individual loss functions as per equation 4.5:

$$
L_{overall} = L_c + L_{JSD} + L_{desc} + L_{fm\_norm} \tag{4.5}
$$

4.5 Implementation Details

We use a pre-trained Densenet-121 model with USAM block after stage 1 of the network and fine-tune it on our custom dataset for 20 epochs. We take a batch size of 16, i.e., eight pairs of images, each consisting of two images of the same identity. The initial learning rate is 10^{-3} with stochastic gradient descent and MultiStepLR optimizer. The mean and standard deviation of the norm of the output attention feature map using the pre-trained model is 38.568 and 17.381, respectively, as used in equation 4.4.

While training the model incorporating descriptor loss, we observed that model was driving the keypoint attention feature map to zero leading to no keypoint detection. It is a trivial solution; with no keypoints, no descriptors would exist, leading to zero descriptor loss. To avoid this, we added a keypoint feature map regularizer term we found to be insufficient making it difficult to extract keypoints. The network was not able to drive this loss to zero. Thus, we are yet to use keypoint matching similarity during our evaluation.

4.6 Evaluation Metrics & Protocols

We have used the following metrics and standard bio-metric assessment protocols to compare our methodology with other baselines:

4.6.1 Evaluation Metrics

These metrics are evaluated on the full test sets and include:

1. **mAP:** Mean Average Precision (mAP) is the mean of AP (average precision) over all classes. Here, AP is the weighted mean of precisions at different thresholds. The weight is an increase in recall as compared to the previous threshold. The formula for AP is given below:
   $AP = \sum(R_n - R_{n-1})P_n$
2. **Top-1 accuracy:** The proportion of examples for which the predicted label matches the single target label. It is also called Rank-1 accuracy.
3. **Top-5 accuracy:** It considers a classification correct if any of the five predictions match the target label and is also known as Rank-5 accuracy.

4.6.2 Evaluation Protocols

These are standard bio-metric assessment protocols, commonly used in the evaluation of human face recognition algorithms.

1. **Close-set Identification:** The test set generates probe and gallery sets by 100 random trials. Here, the identities of the images in the probe set are present in the gallery set. The number of images per identity is set to 4 in our experiments. The identity of the probe image is chosen by comparing its cosine similarity score over the entire gallery set, and the one with the maximum value is selected.
2. **Open-set Identification:** The test set in the open set is the same as the closed set, but the probe set has some identities(odd-numbered) which are not present in the gallery setting. Hence ideally should be classified as unknown. DIR(Detection Identification Rate) with varying FAR(False acceptance rate) percentages are computed as a metric for this protocol. FAR is an estimate of the probability that an alarm is incorrectly sounded on an individual not present in the database of biometric/gallery.
3. **Verification:** Verification is very crucial to confirm whether two images belong to the same individual or not. Positive and negative scores are calculated for all the samples in the test set. The maximum similarity score of a probe image with an image of the same class in the gallery set is defined as a positive score. In contrast, negative scores are the maximum similarity scores for the other classes (except their class). TAR (True acceptance rate) with varying FAR values is reported as a metric for this protocol.

4.7 Experimental Results

Figure 4.9 shows a sample image from the ATRW dataset and its keypoint feature map, which is further used to detect the top 20 keypoints on the image. It also shows how the image and feature map change after applying homography. The USAM output channel feature map with 64 channels is also shown, with each channel capturing essential information.

Table 4.2 displays the mAP, top-1, and top-5 accuracy of the ATRW dataset. Here, 'single-cam' refers to the set of images where the identity only appears in one camera and 'cross-cam' where the target appears in multiple cameras. We have used full-test set for the evaluation. Using JS-Divergence (Model: *Our(JSD)*) instead of KL improves the performance by 2%, 1% and 2% for average mAP, top-1 and top-5 accuracy. Further, using USAM block (Model: *Our(JSD+USAM w/o Desc loss)*) further improves the results by nearly 2% for all the metrics.

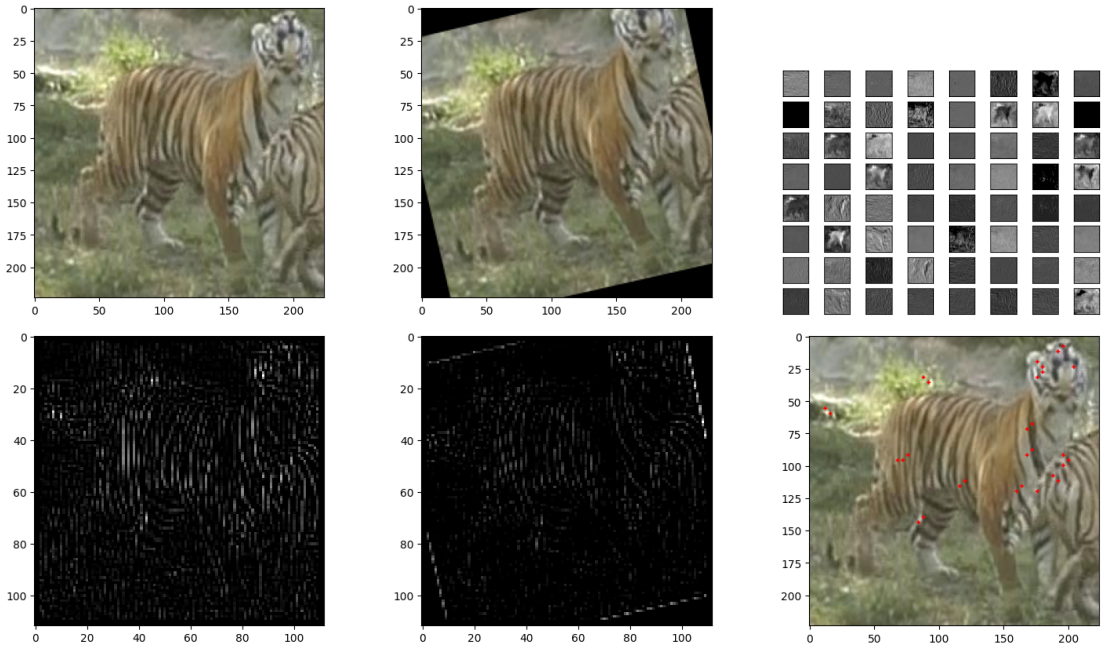


Figure 4.9: Sample image from ATRW dataset, its homography, USAM output channel feature maps, the keypoint feature map generated for original image as well its homography and top-20 keypoints generated

Table 4.3 compares the closed-set rank-1 & rank-5, open-set 1% FAR & 10% FAR, verification 1% FAR on the ATRW test set. All the approaches have comparable performance but our model shows slight improvement by 1% for close-set Rank-5 and verification 1% FAR in comparison to others.

Table 4.4 shows the closed-set rank-1 & rank-5, open-set 1% FAR & 10% FAR, verification 1% FAR on the Leopard test set. We observe that using JS-Divergence improves the results as compared to KL-divergence for the Leopard dataset too. Further, using USAM block along with JS-Divergence improves top-5 close-set identification by 1%.

Figure 4.10 displays samples with top 20 keypoints chosen by our model and their corresponding keypoint attention feature maps. We can see that the model is trying to capture the flank information in the attention feature map.

Figure 4.11 shows the output keypoint feature map with descriptor loss but no feature map regularizer. We can see that the descriptor loss drives the feature map values to zero as the network is trained.

Figure 4.12 shows the visualization of keypoint feature map after adding the feature map regularizer. We observe that it tries to retain the feature map values but the network

Model	mAP			Top-1			Top-5		
	Single-cam	**Cross-cam**	**Aver-age**	**Single-cam**	**Cross-cam**	**Aver-age**	**Single-cam**	**Cross-cam**	**Aver-age**
Cross-Entropy	0.746	0.484	0.615	0.894	0.816	0.855	0.956	0.925	0.9405
PFID(KL) [45]	0.7565	0.6130	0.6848	0.9028	0.7428	0.8228	0.9628	0.8514	0.9071
Our(JSD)	0.7805	0.6309	0.7057	0.9142	0.76	0.8371	0.9685	0.8857	0.9271
Our(JSD+ USAM w/o Desc loss)	0.7703	0.6861	0.7282	0.8914	0.8228	**0.8571**	0.9685	0.9428	**0.9556**
Our(JSD+ USAM with Desc loss)	0.784	0.6839	**0.7339**	0.9171	0.7885	0.8528	0.9714	0.9314	0.9514

Table 4.2: Results comparing classification mAP, top-1 and top-5 accuracy on the test set of *Amur Tigers(ATRW)* with ICCV settings.

Model	Close-set		Open-Set		Verification
	Rank-1	**Rank-5**	**1% FAR**	**10% FAR**	**1% FAR**
Cross-Entropy	0.8928	0.9578	0.842	0.9688	0.9727
PFID(KL) [45]	**0.9**	0.9585	0.8479	0.9717	0.969
Our(JSD)	0.893	0.9581	**0.8503**	0.9665	0.9716
Our(JSD+USAM w/o Desc loss)	0.8916	0.9629	0.8415	0.971	**0.9846**
Our(JSD+USAM with Desc loss)	0.8916	**0.9635**	0.8502	**0.9729**	0.9818

Table 4.3: Results comparing closed-set rank-1 & rank-5, open-set 1% FAR & 10% FAR, verification 1% FAR on the test set of *Amur Tigers(ATRW)*

Model	Close-set		Open-Set		Verification
	Rank-1	**Rank-5**	**1% FAR**	**10% FAR**	**1% FAR**
Cross-Entropy	0.5214	0.7173	0.4465	0.7971	0.7977
PFID(KL) [45]	0.5225	0.7037	0.4512	0.7956	0.8025
Our(JSD)	**0.536**	0.7412	**0.5018**	**0.82**	**0.8252**
Our(JSD+USAM w/o Desc loss)	0.5237	**0.7576**	0.47	0.8184	0.8
Our(JSD+USAM with Desc loss)	0.5259	0.7564	0.472	0.817	0.798

Table 4.4: Results comparing classification closed-set rank-1, open-set rank-1, verification 1% FAR on the test set of *Leopard dataset*

fails to drive this loss to zero, it remains constant over epochs.

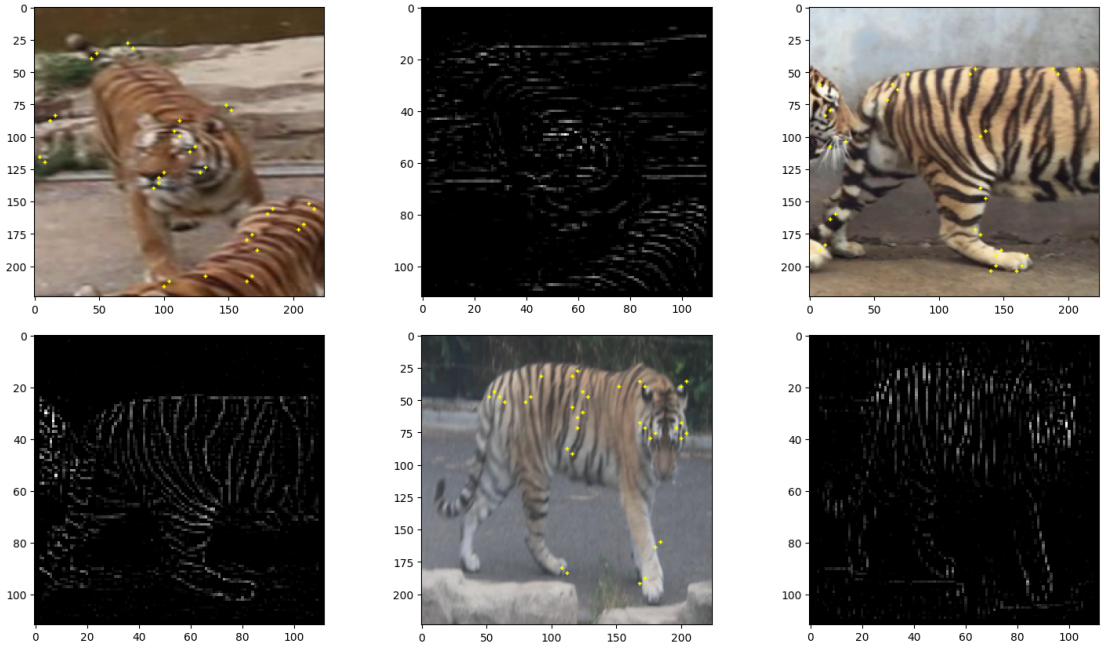
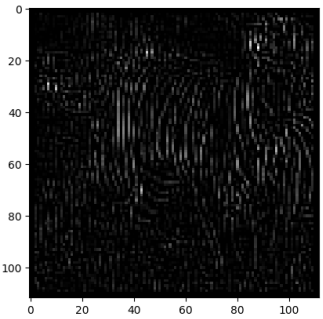


Figure 4.10: Sample image from the ATRW dataset with their keypoint attention feature map

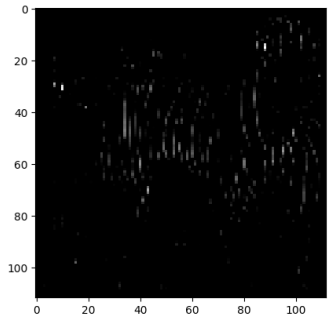
Figure 4.13 shows samples from the Leopard dataset with their keypoint attention feature map trying to capture the rosette information. We get a poor feature map for the last image, which is blurred.

4.7.1 Misclassification scenarios

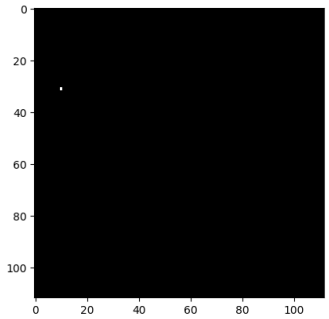
A query image is assigned an ID based on the ID of a maximally similar image predicted by the model. From Figure 4.14 to 4.22, we try to capture misclassification patterns and model confusion through different test samples from both datasets. The first column shows the query image, the middle one shows the best match predicted by our model, and the last is a ground-truth similar image. The misclassification patterns include poor image quality/occlusions, lighting and pose fluctuations, and fine-grained images. Through these samples, we can see that the re-identification images pose very subtle differences which are challenging to identify even by humans.



(a) Output at Batch 0 epoch 1

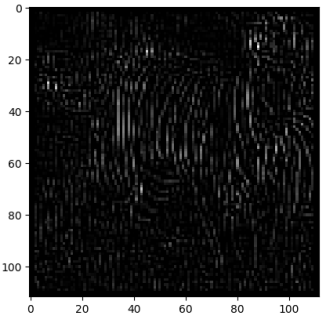


(b) Output at Batch 7 epoch 1

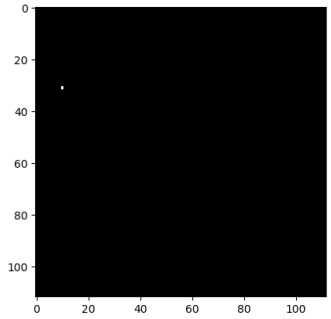


(c) Output at Batch 20 epoch 1

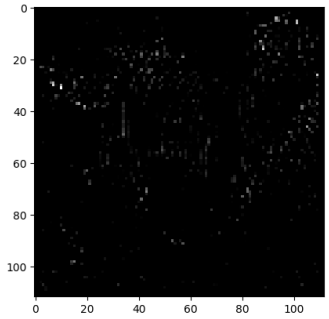
Figure 4.11: Keypoint feature map with descriptor loss without feature map regularizer



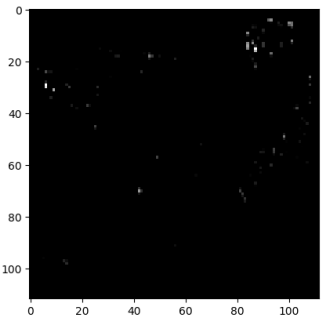
(a) Output at Batch 0 epoch 1



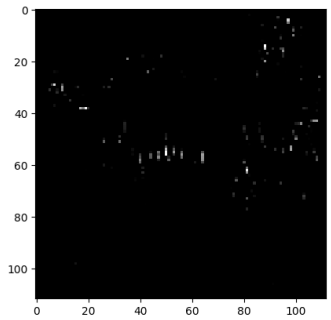
(b) Output at Batch 8 epoch 1



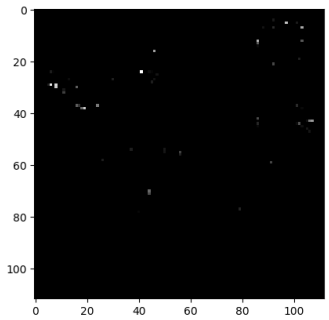
(c) Output at Batch 14 epoch 1



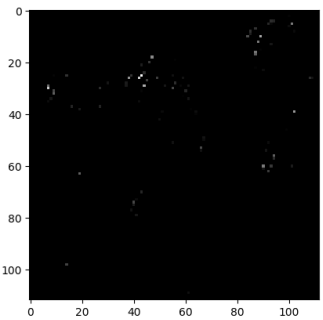
(d) Output at Batch 27 epoch 1



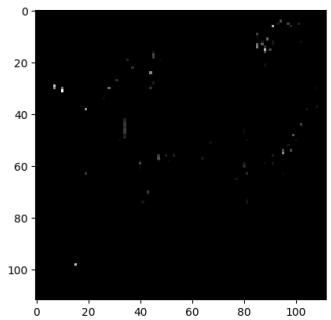
(e) Output at Batch 49 epoch 1



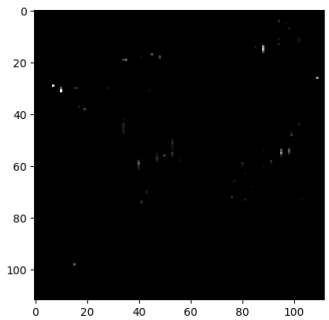
(f) Output at Batch 77 epoch 1



(g) Output at Batch 103 epoch 1



(h) Output at Batch 133 epoch 1



(i) Output at Batch 161 epoch 1

Figure 4.12: Keypoint feature map with descriptor loss without feature map regularizer

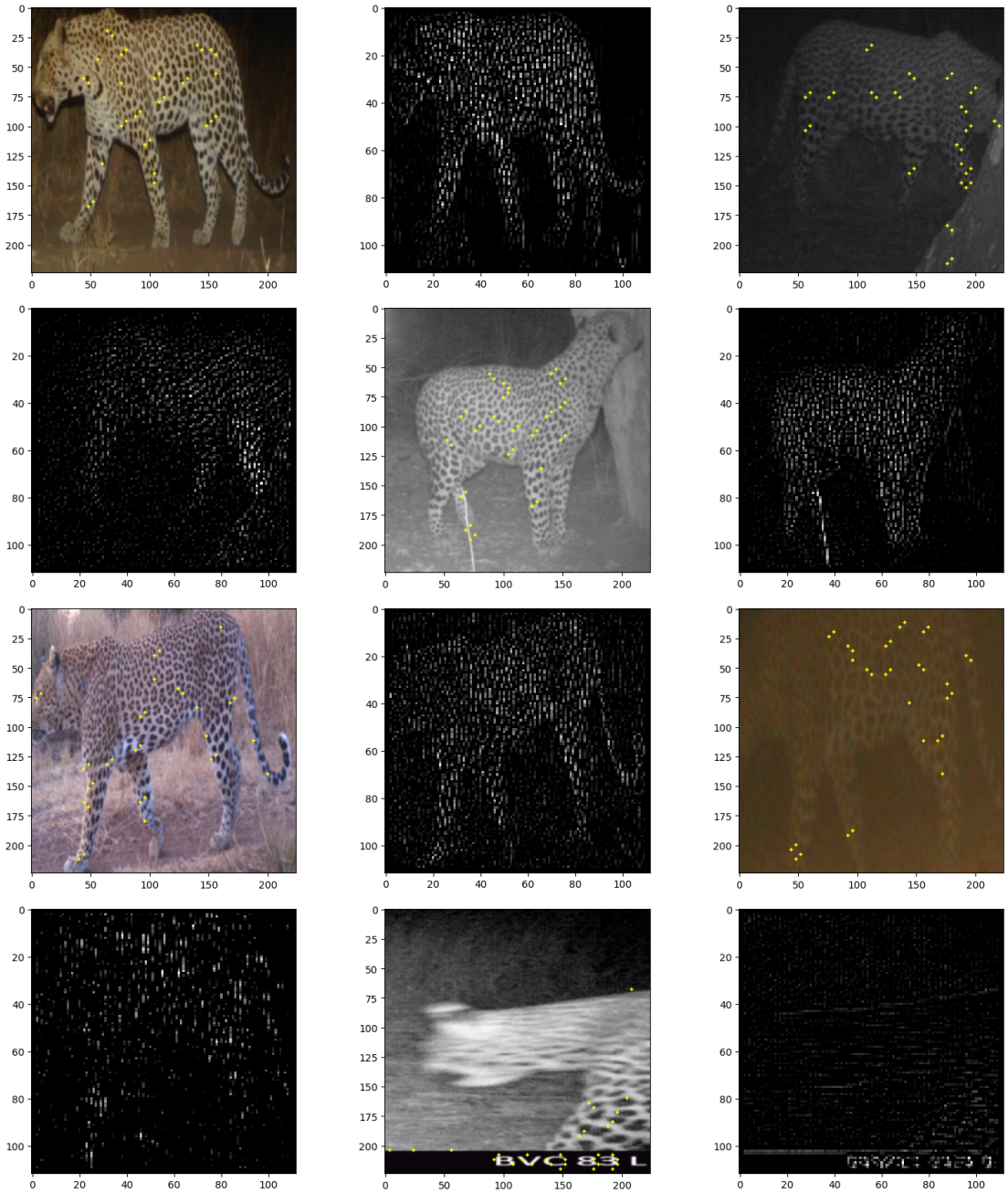


Figure 4.13: Sample image from the Leopard dataset with their keypoint attention feature map

Figure 4.14: Incorrect identification: Poor image quality

Figure 4.15: Incorrect identification: Lighting fluctuation

Figure 4.16: Incorrect identification: Fine-grained images

Figure 4.17: Incorrect identification: Pose variation

Figure 4.18: Misclassified sample from the probe set with its incorrect match found by our model and its correct match from the gallery set of the ATRW dataset

Figure 4.19: Incorrect identification: Poor image quality

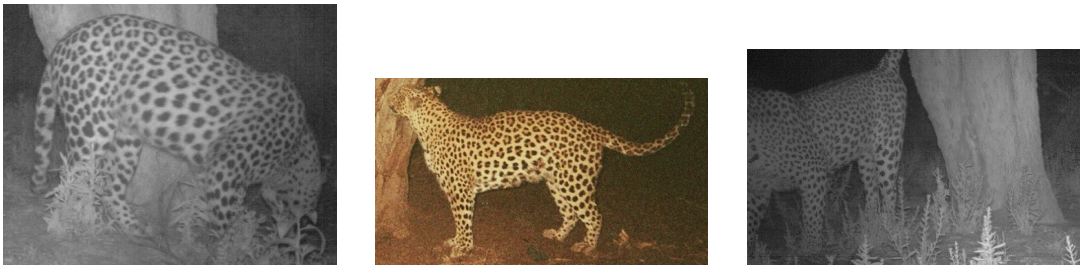


Figure 4.20: Incorrect identification: Lighting fluctuation

Figure 4.21: Incorrect identification: Occlusions

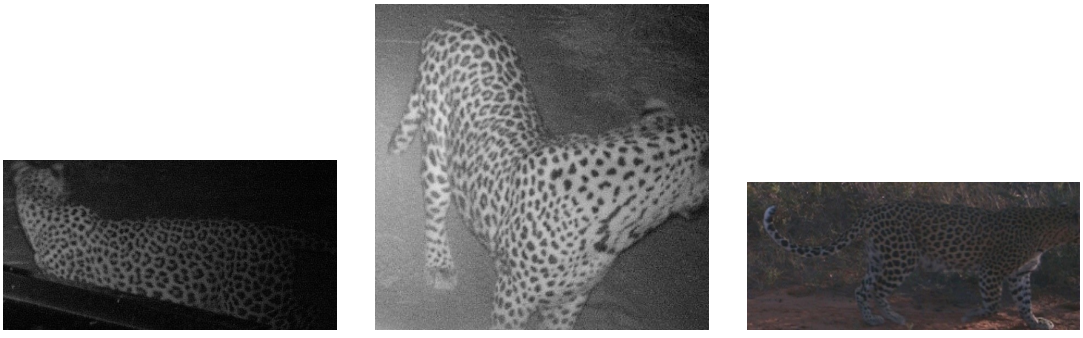


Figure 4.22: Incorrect identification: Pose variation

Figure 4.23: Misclassified sample from the probe set with its incorrect match found by our model and its correct match from the gallery set of the Leopard dataset

CHAPTER 5

Conclusion & Future Work

5.1 Conclusion

In conclusion, this thesis has demonstrated the potential of using deep learning models for animal detection and re-identification tasks. We have used of state-of-the-art model, YOLOv5, for animal detection with a reasonable accuracy, while explored keypoint infused approach for individual animal re-identification.

Our model got overall 95% accuracy on the test data and excellent performance on important and endangered species like tiger, leopard and hyena. Additionally, our tool, CaTRAT, facilitates seamless animal detection and segregation of images into respective folders. This was used by WII for processing the camera trap images for AITE 2022.

Further, we explored the keypoint based approach for animal re-identification. Our network performs is evaluated on different standard biometric evaluation protocols and is able to detect important keypoints in an unsupervised setting. However, there is still much to be explored in this field.

5.2 Future Work

For animal detection, we can work on below items moving forward:

1. We can add more data for classes with fewer samples to create a more balanced dataset and try to reduce its limitations.
2. Blank images don't contribute to the training loss and are observed to often confuse with other classes. We can work on improving the blank detection.
3. We can further improve the overall performance on detection with more confident predictions and use incremental learning for training in the upcoming cycles.

For animal re-identification, we can work on below items:

1. We can include keypoint matches between query and gallery set images as a similarity criterion between them.
2. We can explore using graph neural networks for keypoint matching.

With further advancement of AI, we hope that it continues to contribute in protecting and preserving wildlife in the years to come.

REFERENCES

[1] **Azizpour, H.**, **A. S. Razavian**, **J. Sullivan**, **A. Maki**, and **S. Carlsson** (2015). Factors of transferability for a generic convnet representation. *IEEE transactions on pattern analysis and machine intelligence*, **38**(9), 1790–1802. 8

[2] **Beery, S.**, **D. Morris**, and **S. Yang** (2019). Efficient pipeline for camera trap image review. *arXiv preprint arXiv:1907.06772*. 3, 8, 16

[3] **Castel, E.** (2002). Panthers, leopards and cheetahs. notes on identification. *Trabajos de Egiptología*, **1**, 17–28. 5

[4] **Cui, Y.**, **Y. Song**, **C. Sun**, **A. Howard**, and **S. Belongie**, Large scale fine-grained categorization and domain-specific transfer learning. *In Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018. 8

[5] **Dalal, N.** and **B. Triggs**, Histograms of oriented gradients for human detection. *In 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1. Ieee, 2005. 7

[6] **Deb, D.**, **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)*. IEEE, 2018. 4, 9

[7] **Deng, J.**, **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*. Ieee, 2009. 4

[8] **DeTone, D.**, **T. Malisiewicz**, and **A. Rabinovich**, Superpoint: Self-supervised interest point detection and description. *In Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2018. 11

[9] **Donahue, J.**, **Y. Jia**, **O. Vinyals**, **J. Hoffman**, **N. Zhang**, **E. Tzeng**, and **T. Darrell**, Decaf: A deep convolutional activation feature for generic visual recognition. *In International conference on machine learning*. PMLR, 2014. 8

[10] **Dusmanu, M.**, **I. Rocco**, **T. Pajdla**, **M. Pollefeys**, **J. Sivic**, **A. Torii**, and **T. Sattler**, D2-net: A trainable cnn for joint detection and description of local features. *In CVPR 2019-IEEE Conference on Computer Vision and Pattern Recognition*. 2019. 11

[11] **Everingham, M.**, **L. Van Gool**, **C. K. Williams**, **J. Winn**, and **A. Zisserman** (2009). The pascal visual object classes (voc) challenge. *International journal of computer vision*, **88**, 303–308. 7

[12] **Felzenszwalb, P. F.**, **R. B. Girshick**, **D. McAllester**, and **D. Ramanan** (2009). Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, **32**(9), 1627–1645. 7

[13] **Felzenszwalb, P. F.**, **R. B. Girshick**, **D. McAllester**, and **D. Ramanan** (2010). Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, **32**(9), 1627–1645. 8

[14] **Girshick, R.**, Fast r-cnn. *In Proceedings of the IEEE international conference on computer vision*. 2015. 7

[15] **Girshick, R.**, **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. 7

[16] **He, K.**, **G. Gkioxari**, **P. Dollár**, and **R. Girshick**, Mask r-cnn. *In Proceedings of the IEEE international conference on computer vision*. 2017. 7

[17] **Hiby, L.**, **P. Lovell**, **N. Patil**, **N. S. Kumar**, **A. M. Gopalaswamy**, and **K. U. Karanth** (2009). A tiger cannot change its stripes: using a three-dimensional model to match images of living tigers and tiger skins. *Biology letters*, **5**(3), 383–386. 4, 5

[18] **Huang, G.**, **Z. Liu**, **L. Van Der Maaten**, and **K. Q. Weinberger**, Densely connected convolutional networks. *In Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017. 8

[19] **Jebreel, N. M.**, **J. Domingo-Ferrer**, **D. Sánchez**, and **A. Blanco-Justicia** (2021). Keynet: An asymmetric key-style framework for watermarking deep learning models. *Applied Sciences*, **11**(3), 999. 11

[20] **Krizhevsky, A.**, **I. Sutskever**, and **G. E. Hinton** (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, **25**. 8

[21] **Law, H.** and **J. Deng**, Cornernet: Detecting objects as paired keypoints. *In Proceedings of the European conference on computer vision (ECCV)*. 2018. 8

[22] **Lin, J.**, **Z. Zheng**, **Z. Zhong**, **Z. Luo**, **S. Li**, **Y. Yang**, and **N. Sebe** (2022). Joint representation learning and keypoint detection for cross-view geo-localization. *IEEE Transactions on Image Processing*, **31**, 3780–3792. 11, 28, 30, 31

[23] **Lin, T.-Y.**, **P. Dollár**, **R. Girshick**, **K. He**, **B. Hariharan**, and **S. Belongie**, Feature pyramid networks for object detection. *In Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017. 7

[24] **Lin, T.-Y.**, **P. Goyal**, **R. Girshick**, **K. He**, and **P. Dollár**, Focal loss for dense object detection. *In Proceedings of the IEEE international conference on computer vision*. 2017. 7, 8

[25] **Liu, C.**, **R. Zhang**, and **L. Guo**, Part-pose guided amur tiger re-identification. *In Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*. 2019. 4, 10

[26] **Liu, N.**, **Q. Zhao**, **N. Zhang**, **X. Cheng**, and **J. Zhu**, Pose-guided complementary features learning for amur tiger re-identification. *In Proceedings of the IEEE/CVF international conference on computer vision workshops*. 2019. 4, 9, 10

[27] **Liu, W.**, **D. Anguelov**, **D. Erhan**, **C. Szegedy**, **S. Reed**, **C.-Y. Fu**, and **A. C. Berg**, Ssd: Single shot multibox detector. *In European conference on computer vision*. Springer, 2016. 7, 8

[28] **Lowe, D. G.** (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, **60**(2), 91–110. 7, 11

[29] **Moskvyak, O.**, **F. Maire**, **F. Dayoub**, and **M. Baktashmotlagh**, Keypoint-aligned embeddings for image retrieval and re-identification. *In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2021. 10

[30] **Mukherjee, D.**, **Q. Jonathan Wu**, and **G. Wang** (2015). A comparative experimental study of image feature detectors and descriptors. *Machine Vision and Applications*, **26**(4), 443–466. 11

[31] **Ono, Y.**, **E. Trulls**, **P. Fua**, and **K. M. Yi** (2018). Lf-net: Learning local features from images. *Advances in neural information processing systems*, **31**. 11

[32] **Oza, P.** and **V. M. Patel**, C2ae: Class conditioned auto-encoder for open-set recognition. *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019. 9

[33] **Pedersen, M.**, **J. B. Haurum**, **T. B. Moeslund**, and **M. Nyegaard**, Re-identification of giant sunfish using keypoint matching. *In Proceedings of the Northern Lights Deep Learning Workshop*, volume 3. 2022. 10

[34] **Qian, X.**, **Y. Fu**, **Y.-G. Jiang**, **T. Xiang**, and **X. Xue**, Multi-scale deep learning architectures for person re-identification. *In Proceedings of the IEEE International Conference on Computer Vision*. 2017. 4

[35] **Ravoor, P. C.** and **T. Sudarshan** (2020). Deep learning methods for multi-species animal re-identification and tracking–a survey. *Computer Science Review*, **38**, 100289. 8

[36] **Redmon, J.**, **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. 3, 7, 8, 16

[37] **Ren, S.**, **K. He**, **R. Girshick**, and **J. Sun** (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, **28**. 7

[38] **Rosten, E.** and **T. Drummond**, Machine learning for high-speed corner detection. *In European conference on computer vision*. Springer, 2006. 3, 11

[39] **Rublee, E.**, **V. Rabaud**, **K. Konolige**, and **G. Bradski**, Orb: An efficient alternative to sift or surf. *In 2011 International conference on computer vision*. Ieee, 2011. 11

[40] **Russakovsky, O.**, **J. Deng**, **H. Su**, **J. Krause**, **S. Satheesh**, **S. Ma**, **Z. Huang**, **A. Karpathy**, **A. Khosla**, **M. Bernstein**, *et al.* (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision*, **115**(3), 211–252. 4, 8

[41] **Sarlin, P.-E.**, **D. DeTone**, **T. Malisiewicz**, and **A. Rabinovich**, Superglue: Learning feature matching with graph neural networks. *In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020. 11

[42] **Schneider, S.**, **G. W. Taylor**, and **S. Kremer**, Deep learning object detection methods for ecological camera trap data. *In 2018 15th Conference on computer and robot vision (CRV)*. IEEE, 2018. 8

[43] **Schumann, A.** and **R. Stiefelhagen**, Person re-identification by deep learning attribute-complementary information. *In Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2017. 4

[44] **Sermanet, P.**, **D. Eigen**, **X. Zhang**, **M. Mathieu**, **R. Fergus**, and **Y. LeCun** (2013). Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*. 8

[45] **Shukla, A.**, **G. S. Cheema**, **S. Anand**, **Q. Qureshi**, and **Y. Jhala**, Primate face identification in the wild. *In Pacific Rim International Conference on Artificial Intelligence*. Springer, 2019. 9, 35

[46] **Shukla, A.**, **G. Sigh Cheema**, **P. Gao**, **S. Onda**, **D. Anshumaan**, **S. Anand**, **R. Farrell**, *et al.*, A hybrid approach to tiger re-identification. *In Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*. 2019. 4, 5, 9

[47] **WII** (2022). Status of tigers 2022 summary. https://wii.gov.in/images//images/documents/publications/Status_Tigers_2022_Summary.pdf. 1, 2, 4, 6

[48] **Yosinski, J.**, **J. Clune**, **Y. Bengio**, and **H. Lipson** (2014). How transferable are features in deep neural networks? *Advances in neural information processing systems*, **27**. 8

[49] **Yu, J.**, **H. Su**, **J. Liu**, **Z. Yang**, **Z. Zhang**, **Y. Zhu**, **L. Yang**, and **B. Jiao**, A strong baseline for tiger re-id and its bag of tricks. *In Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*. 2019. 4, 5, 10

[50] **Zheng, L.**, **Y. Yang**, and **A. G. Hauptmann** (2016). Person re-identification: Past, present and future. *arXiv preprint arXiv:1610.02984*. 4