University of Tartu

Faculty of Science and Technology

Institute of Technology

Ilja Pavlovs

Animal recognition using deep learning

Bachelor Thesis (12 ECTS)
Curriculum Science and Technology

Supervisors:

Egils Avots, MSc
Prof. Gholamreza Anbarjafari

Tartu 2021

Abstract

Animal recognition using deep learning

Camera traps are widely used for wildlife monitoring. In this work machine learning based data processing pipeline is assembled for animal detection on the camera trap images focusing on the ungulate species. The typical animal detection challenges are noted, and available solutions are evaluated. As the result of this work, two different deep neural networks Faster R-CNN and RetinaNet were trained, achieving 0.2786 mAP@0.5:0.05:0.95 and 0.4562 mAP@0.5 on the dataset of interest gathered in the Latvian forest regions during the "ICT-based wild animal census approach for sustainable wildlife management" project. Additionally, different learning optimization techniques such as data augmentation and oversampling were implemented and assessed.

CERCS: T111 Imaging, image processing

Keywords: Computer Vision, Machine Learning, Animal detection

Loomade tuvastamine süvaõppega

Kaameralõkse kasutatakse laialdaselt elusloodu seireks. Selles töös moodustati masinõppepõhine süsteem loomade tuvastamiseks kaameralõksude fotodes keskendudes sõralistele. Täheldati mitmeid varasemalt teadaolevaid probleeme loomatuvastuses ja hinnati tehnilisi lahendusi nende leevendamiseks. Töö tulemusena treeniti kaks sügavat närvivõrku: Faster R-CNN ja RetinaNet, mis saavutasid vastavalt 0,2786 mAP @ 0,5:0,05:0,95 ja 0,4562 mAP @ 0,5. Mudelid treeniti Läti metsadest kogutud andmestikul mis on osaks projektist IKT-põhine metsloomade loendamine jätkusuutlikuks looduse majandamiseks". Lisaks rakendati ja hinnati erinevaid õppe optimeerimise meetodeid, nagu andmete rikastamine ja üleproovide võtmine.

CERCS: T111 Pilditehnika

Märksõnad: raalnägemine, masinõpe, loomatuvastus

Acknowledgments

I would like to thank my supervisors Prof. Gholamreza Anbarjafari and Egils Avots, for their support and guidance throughout the writing of this thesis.

Table of Contents

List of Figures

List of Tables

Abbreviations and Acronyms

EU - European Union

ML - Machine learning

DNN - Deep neural network

SGD - Stochastic gradient descent

CNN - Convolution neural network

FLOPS - Floating point operations per second

SSD - Single Shot Detector

RPN - Region Proposal Network

R-CNN - Region Based Convolutional Neural Network

FPN - Feature Pyramid Network

mAP - Mean average precision

IoU - Intersection over union

1 Introduction

The rapid expansion of ungulates has been observed in many European forest regions since the 2000s as the upshot of a complex of factors including decreased hunting activity and changes in legislation regarding poaching and abandonment of land [4]. Although ungulates as natural inhabitants of the forest region usually benefit the forest ecosystem by improving germination conditions for seeds and affecting the forest development, evidence suggests that excessive ungulate density leads to ecosystem disturbance, forestry damage, and disease propagation acceleration [5]. European Wilderness Society reports that annual damage caused by ungulates in the agricultural and forestry sector in EU countries exceeds 100 million euros, suggesting that this problem already causes significant economic losses and should be addressed. Additionally, the overabundance of the ungulates increases the danger on the highways and is still an unresolved problem as there are on average 750,000 vehicle collisions with ungulates per year in Europe [6]. As part of the solution to these problems, ungulate population control strategies and monitoring systems should be developed and studied. Monitoring systems should estimate ungulate population distribution, dynamics, ecological change indicators and manage ungulate overabundance efficiently and systematically. One of the most common animal monitoring approaches is the camera trap system which will be reviewed in detail.

1.1 Problem Overview

Since the release of the first motion-triggered cameras, i.e., camera traps on the market in 1988, they have been increasingly used for ecological research [7]. Inconspicuous camera traps allowed researchers to gather the wildlife activity insights in its natural habitat unobtrusively. Such a way of data collection allows the data to be collected continuously, non-intrusively, less expensively and gathers the data in hardly accessible places. Obtained data in the form of images or videos need manual review to extract biologically representative information (e.g., animal species, count, behavior, age, and recognition of animal individuals). Researchers then use this data to investigate animal population distribution, their interaction with local ecosystems, intra-community interactions, their dynamics, and other biological parameters related to the research subject. Such fieldwork consumes an unreasonably immense amount of human resources and time, which facilitates inefficient human resource distribution and necessitates

work with smaller data volumes.
This work focuses on the development of the deep neural network-driven data processing pipeline for animal detection on the camera trap videos gathered during the ”ICT-based wild animal census approach for sustainable wildlife management” project, which takes place in Latvia and focuses on the four dominant even-toed ungulate species: red deer (*Cervus elaphus*), roe deer (*Capreolus capreolus*), elk (*Alces alces*) and wild boar (*Sus scrofa*).

1.2 Goals

This thesis aims to train a deep neural network that can detect animal species on the camera trap videos. Considering current progress achieved by similar works and individual aspects of this particular problem, the following tasks are also implied to be achieved:

1. Implement effective DNN learning strategies to compensate for the small and unbalanced dataset.
2. Assess the reliability and speed of different machine learning models.
3. Harness the advantages of the video captures.

1.3 Structure

- Chapter 2 gives an overview of trap camera usage and their limitations in wildlife monitoring.
- Chapter 3 of this thesis reviews similar works and outlines the common detection challenges and their solutions.
- Chapter 4 describes general machine learning concepts and methods used in the thesis.
- Chapter 5 gives an in-depth view of the data used in this work and its pre-processing.
- Chapter 6 presents the results of the work.
- Chapter 7 contains a discussion of the results.
- Chapter 8 concludes the results of the thesis and proposes future improvements.

2 Wildlife monitoring

2.1 Trap cameras

Camera trap (e.g., Figure 2.1) is the camera equipped with the motion sensor (passive or active infrared sensor), which triggers in the presence of motion within the sensor's field of view. Additionally, some cameras are equipped with infrared flash, which allows them to capture animals at night-time. Despite the numerous advantages, trap cameras have limited visibility compared with alternative monitoring approaches such as aerial drone monitoring. Thus, many camera traps are usually required to cover the terrain of the interest, and different camera trap distribution approaches should be considered to gather unbiased data to estimate population indices and other quantitative parameters. This thesis uses available datasets and recordings of the camera traps placed across the Latvian Jaunaklsnava and Kalsnava regions.

Figure 2.1: Camera trap used for gathering the data for the"ICT-based wild animal census approach for sustainable wildlife management" project.

3 Literature Review

During the last ten years, computer vision applications for animal detection is an object of active research. In parallel with increasing concern about rare animal species extinction and availability of high computational power, many novel solutions and approaches were investigated and applied, achieving better and better results. Common challenges and features that complicate precise animal detection were derived from the previous works. They are the following:

1. Falsely triggered captures. Captures that do not contain animals usually take up a large proportion of the data (76% for Snapshot Serengeti dataset [8]).

2. Unbalanced dataset. Due to the differences in animal species population distributions in the wild, some species are captured more frequently than others, which impedes unbiased learning (rare species tend to be miss-classified).

3. Background learning. Camera traps are static, which results in the high similarity of obtained images. Due to the different factors, animal species are unevenly distributed across different camera locations making relations of background and species present, which lowers the model's generality. [9]

In order to address these issues, discover other occurring problems and understand the progress of state-of-the-art solutions to animal detection problems, similar works were investigated.

3.1 Similar works

Mohammed et al. [10] describes a DNN (deep neural network) that can classify the animals performing on the same level as a crowd-sourced team of volunteers (with confidence threshold maintaining most of the data labeled). The classification was subdivided into two subsequent modules: detecting an image containing an animal (VGG is selected as best performing model) and animal classification (Ensemble of models). Such architecture efficiently addressed the empty capture problem. This network covered animal count, behavior, and basic description of species on the image. Mohammed et al. [10] demonstrates that transfer learning does not improve their model precision significantly, which was associated with the relatively large size of the dataset (Snapshot Serengeti Project dataset used by Mohammed et al. has around 1.2

million labeled capture events of 48 species [8]). This article also assesses accuracy improvement for an unbalanced dataset: weighted loss approach, which assigns a higher cost for rare classes, an oversampling approach which passes the rare class images more frequently during model training and emphasis learning approach, which pass the miss-classified images to the model again during iterative process [11]. However, this work considered information extraction as a classification task without addressing animal localization on the image and had certain limitations such as the inability to consider the captured images with multiple species present. Christin et al. [12] describes results of European animal detection using FasterRCNN with InceptionResNet backbone pre-trained on the Open Images Dataset V4 [13]. Open Images Dataset missed species-specific classification and achieved 93% accuracy for higher taxonomy rank animal detection. Although this work does not examine the model for a species-specific dataset and the development of a custom model, it estimates the results obtained by implementing a pre-trained model for a dataset similar to the one used in this thesis.

Another research describes the ResNet -18 model for classifying animals. It was able to achieve 97.6% top-1 accuracy and more than 99.9% top-5 accuracy, which are the highest accuracy scores obtained on a custom dataset to date of the publication [14]. It also developed a $\sim$3.7 million labeled image dataset (NACTI dataset), which included 27 species across five locations in the United States. Data augmentation techniques were applied to make the model more robust (cropping, horizontal flipping, brightness, and contrast variations). Tabak et al. concluded that daytime captures were more easily classified (1.6% top-1 accuracy improvement) [14]. Mohammed et al. [15] proposes a pipeline to accelerate species identification and counting on the camera trap images harnessing advantages of active deep learning and transfer learning assessing small project data limitation problems. The detection task was partitioned into three steps:

1. Running a pre-trained animal detector (Faster-RCNN) with only one class: "Animal". Results are filtered with a 90% confidence threshold (any detection below 90% confidence considered as false detection). This step enables to detect falsely triggered captures with 91.71% accuracy, count the animals achieving 72.4% and 86.8% accuracy for exact and +/-1 bin count estimation accordingly without requiring to label data. Images are cropped around the object detection reducing redundant information and resized. Background detrimental effect on performance is also noted by [9]. Zhongqi et al. [9] shows that repeating species increase miss-classification of other species because the static background is interpreted as an animal recognition feature. Such bias can be avoided by firstly finding bounding boxes and then detecting animals inside them.

2. Embedding the cropped images using a trained deep neural network. Mohammed et al. [15] exploits the ResNet-50 model to learn embedding by classifying the images or using triple loss on the same dataset. Mohammed et al. [15] demonstrated higher accuracy

of triplet loss features, achieving 91.37% compared with 85.23% accuracy for features from the last image classification model layer for 25,000 labels.

3. Active learning phase. Active learning proceeds in the following way: oracle, i.e., intelligent agent labels 1000 random images, which are used for initial classification model training. After that, the algorithm cycles across the unlabeled data selecting samples that oracle is asked to label. Periodically classification model and embedding model are fine-tuned, harnessing labeled data present, and the process repeats until meeting a termination condition. Different sample selection strategies for oracle labeling are implemented, and the k-Center strategy is reported to be the most successful obtaining 92.2% accuracy with 25,000 labels.

Such pipeline architecture allows annotating reliable training datasets, significantly reducing manual annotation time. Although this thesis does not aim to implement active deep learning, Mohammed et al. [15] highlights the different embedding learning approaches and task segmentation advantages.

It is also notable that video format advantages could be harnessed to improve model accuracy further. Instead of applying a classifier to each separate frame, temporal information could be used to build or decrease confidence about certain predictions or implement object and background segmentation, facilitating animal detection considering the occluded background, which usually prevails on the camera trap images. In particular, Zhang et al. [16] addresses the animal segmentation problem from the camera trap videos and proposes an iterative, embedded graph cut (IEC) method for proposing regions that potentially contain the animal. The IEC method generates fewer regions than the Selective search used in the Fast-RCNN while still having good object coverage (percentage of ground-truth regions with intersection-over-union (IoU) higher or equal to 0.5 with proposed regions). Joint convolution neural network (CNN) features and histogram of oriented gradient (HOG) encoded with Fisher Vectors features are learned and used to decide whether image patch belongs to background or contains an animal. Additionally, Zhang et al. [16] proposes a cross-frame verification method for improving animal/background classification. For the testing Zhang et al. [16] used their dataset containing over 1 million camera trap images covering 23 animal species. Considering regions with IoU higher or equal to 0.5 overlaps with ground-truth boxes as positive, the proposed system demonstrated an 83.98% average F-score outperforming the Faster-RCNN region proposal 3.5% and YOLO by $\sim$8%. Model was accommodating cluttered and dynamic scenes, including intermittent and low frame-rate videos while demonstrating great segmentation quality and outperforming Selective Search by average image processing time.

The papers mentioned above propose multiple solutions to the challenges listed at the beginning of the literature review and demonstrate the effectiveness of model design decisions. Models that are used in this thesis are explicitly described in the Methodology section of this paper.

4 Methodology

4.1 Machine Learning

Machine learning is a programming paradigm that studies the mathematical models that can improve their performance over certain types of tasks when fed with the data. Different machine learning approaches usually aim at specific tasks and work with corresponding types of data.

1. **Supervised learning** - Input data is labeled with ground truth value or expected output by intelligent agents. It is mostly used for recognition tasks.
2. **Unsupervised learning** - Input data is not labeled. The model learns to find patterns in the data, which can be used for clustering based on the data commonality evaluation or for generative tasks.
3. **Reinforcement learning** - Model is continuously learning while interacting with intelligent agents.

Learning itself refers to the model weights tuning during the process of loss function minimization. The loss function evaluates how well a model performed in response to a particular input. Minimization of the loss function is realized via stochastic gradient descent (SGD) algorithm, which updates model weights so that they change in the direction that contributes to going down the loss function gradient. Learning rate controls the extent to which parameters are tuned in each iteration. SGD in general form demonstrated below:

$$w_n = w_{n-1} - \eta * \nabla(L) \tag{4.1}$$

Where:
w - parameter vector
n - number of iteration
η - learning rate
L - loss function

SGD has different variations, e.g., momentum method where each new parameter update considers previous parameter update and AdaGrad with individual learning rate for each pa-

rameter.

4.1.1 Hyperparameters

Hyperparameters are general parameters that define global characteristics of the model structure and training procedure. Hyperparameters can affect neural network success drastically and should be initialized appropriately. In this work, hyperparameters were taken from similar works. However, it is possible to perform hyperparameter optimization (e.g., Grid Search, Random Search) to find the optimal or good-enough parameters for the model and make the neural network perform better for the task of interest. Tan et al. [17] reported that network architecture design could be optimized by jointly scaling network depth, width, and resolution and produce better classification accuracy.

Training hyperparameters:

- Batch size - defines the number of samples passed to the model before its weights are updated. The bigger batch size makes the loss function gradient estimation less noisy, ensuring that weight update will result in a loss decrease. However, big batch size will reduce stochastic properties of the gradient descent, which can slow down the learning process as more images should be processed before weights are updated and make it less probable for the model to converge to the global minimum when the local minimums or settling points are present.
- Learning rate - controls the model weights update rate.
- Number of epochs - defines the number of times that the learning algorithm works through the entire training data-set.

CNN hyperparameters:

- Network depth - defines the number of the layers that the model has.
- Network width - defines the number of feature maps at each layer.
- Network resolution - defines the resolution of the input.

4.1.2 Artificial Neural Networks

An artificial Neural network (ANN) is a machine learning model consisting of interconnected artificial neurons that can transmit signals in the biological neuron manner. Artificial neurons receive inputs from other neurons, calculate weighted sum (each connection has assigned

to its weight which is tuned during learning), pass it through nonlinear function (e.g., Sigmoid, ReLu), and transmit the signal to the downstream neurons. Artificial neurons also have a weighted input of 1, which is called neuron bias. Neural networks contain many communicating neurons and usually are represented in a layered structure where each layer is interconnected with the next one. "Deep neural networks" usually refer to neural networks which are at least three layers deep and can hierarchically learn features. Such networks can autonomously learn high-level abstract features from the high-dimensional, extensive data, which gives an advantage in working with multidimensional sequential and spatial data, e.g., sound, images.

4.1.3 Convolution Neural Network (CNN)

Convolution Neural Network (CNN) is a deep neural network that is constructed from convolution layers. Convolution layers incorporate convolution operation itself and can contain activation (adding non-linearity) and pooling operations (down-sampling of the image) depending on the model architecture. The last convolution layer output (feature map) is flattened (features are transformed into the 1D array) and fully connected to all possible outputs directly or through intermediate hidden layers, and softmax function is applied to make sure that the output sums up to 1 and thus could be considered as a probability. When such a computation graph is initialized, passing the image through it will result in the vector of probabilities of getting each output. CNN is usually tuned with stochastic gradient descent that aims to minimize cross-entropy loss which is a frequent function for multi-class categorization problems [18]. CNNs turned out to be very effective when working with images because of their ability to effectively extract higher-level semantic features in a bottom-up learning manner resembling our visual cortex working principle.
CNNs are the most used neural networks to solve computer vision challenges. The most common tasks of computer vision (see Figure 4.1) are:

1. Classification. The model output is the class label and confidence score.
2. Classification and localization. Model determines the location of the object by enclosing it into rectangle and classifies the object inside. Model outputs the four coordinates, class label and confidence score.
3. Object detection, i.e., multiple object detection and localization.
4. Instance segmentation. The object boundaries are detected and model outputs masks for the objects (array of coordinates) and associated class label and confidence predictions.

This thesis aims to solve the object detection problem as it should be possible to localize and classify multiple animals in one frame. Object classification can be obtained from the high-level features by passing them through the feed-forward neural network with a fully connected layer at the end with cross-entropy loss. However, bounding box localization requires a more complicated algorithm.

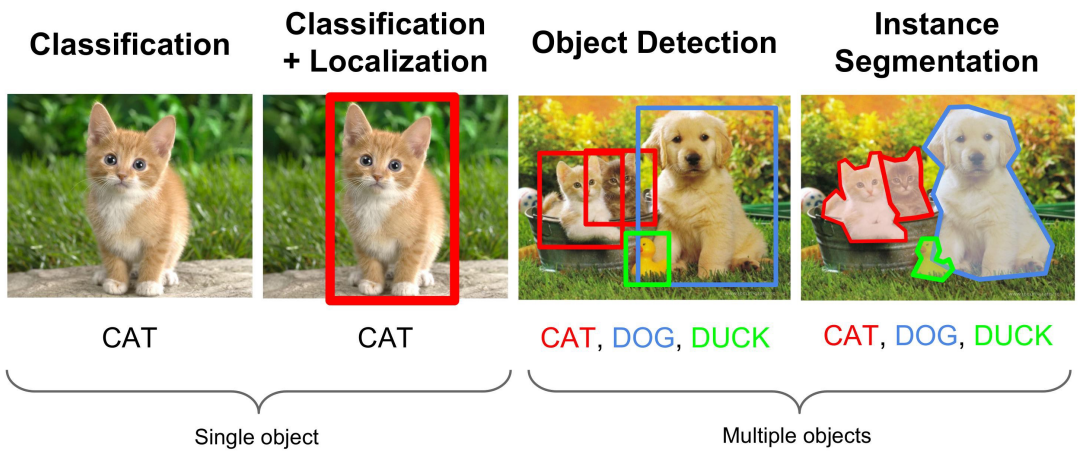

Figure 4.1: CNN computer vision tasks [1]

4.2 Detection neural networks

Currently, there are two general detection approaches: one stage detection and two-stage detection or region proposal detection. As the name suggests, one stage detection represents holistic structure, usually in large sequential CNN (e.g., YOLO, SSD), which generates all predictions by a single run. Whereas two-stage detection divides detection tasks into region proposal stage and region classification stage. One stage detectors are generally faster but less accurate.
The general structure of the detector involves image embedding (obtaining low dimensional image representation), object localization, and classification. Object localization is learned by regression of bounding box coordinates, and classification is learned by minimizing classification loss. The detector model architectures could be generally subdivided into three functional modules:

1. **Backbone network** - DNN consisting from the convolution layers which is used for the feature extraction from the input image. Usually backbone networks which were pre-trained on the natural image dataset (e.g. ImageNet, COCO, Open Images Dataset, CIFAR10) are used. Common networks used as the backbone are: ResNet50 (containing residual connections) [19], VGG16 [20], Inception-ResNet V2 (combination of the Inception and ResNet architectures) [21], and DarkNet-19 [22] used in YOLO.

2. **Neck** - DNN module on the top of the Backbone network. The Neck network is not an essential module that takes and processes inputs from the different layers of the Backbone network, harnessing advantages of data pattern distribution over different feature map scales. Standard Neck networks are FPN (Feature Pyramid Network), BiFPN (Bidirectional Feature Pyramid Network). However, they are not used frequently because of their high computational complexity [23].

3. **Head** - Usually it is a feed-forward neural network which performs classification or re-

gression task. The detector could have multiple heads for performing different classification and regression tasks.

The detector outputs the bounding boxes with corresponding labels and confidence scores. The confidence score is calculated from the classification head last layer's output by applying softmax function, which normalizes the output to probability distribution [24].

The detector can generate redundant, overlapping predictions for the same object and produce

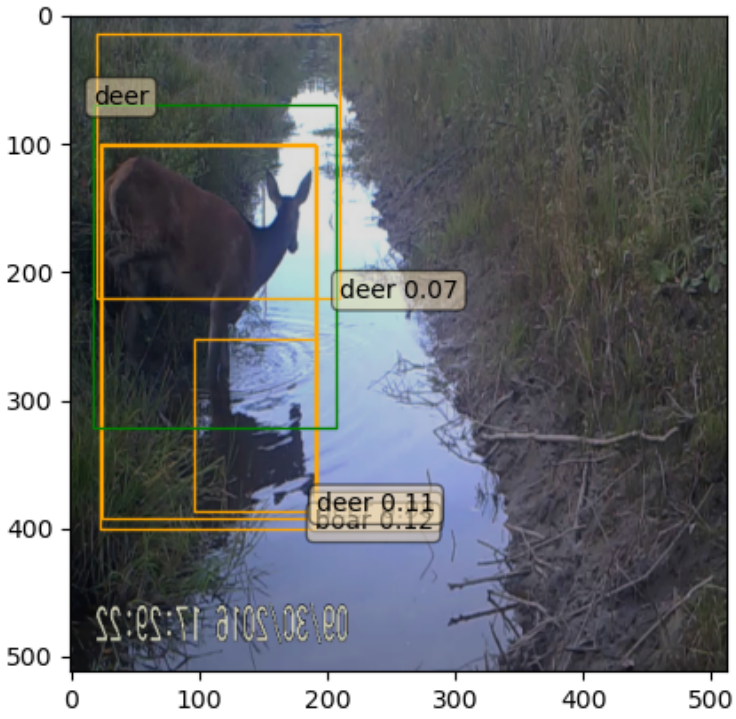

Figure 4.2: Faster R-CNN predictions before non-maximum suppression. Faster R-CNN produces redundant, overlapping bounding boxes and bounding boxes with low confidence scores. The orange rectangles show the model predictions, and the green rectangle shows the ground truth bounding box.

low confidence predictions what can be overcome with **non-maximum suppression** proposal filtering technique. This method uses the IoU metric to measure how accurately objects are superimposed on each other. IoU is calculated by dividing the prediction intersection area with their union area, which produces the result between 0 (predictions do not overlap) and 1 (predictions are perfectly superimposed). If IoU exceeds a certain threshold, the bounding box with a lower confidence score is excluded. Additionally, bounding boxes with small confidence scores could be filtered to exclude miss-classified regions. It was observed in this work that the detector frequently successfully localized the animal but produced both "boar" and "dear" class bounding boxes around the object. In order to address this issue, a species overlapping criterion was added, which excluded the prediction with less confidence score in case of multiple

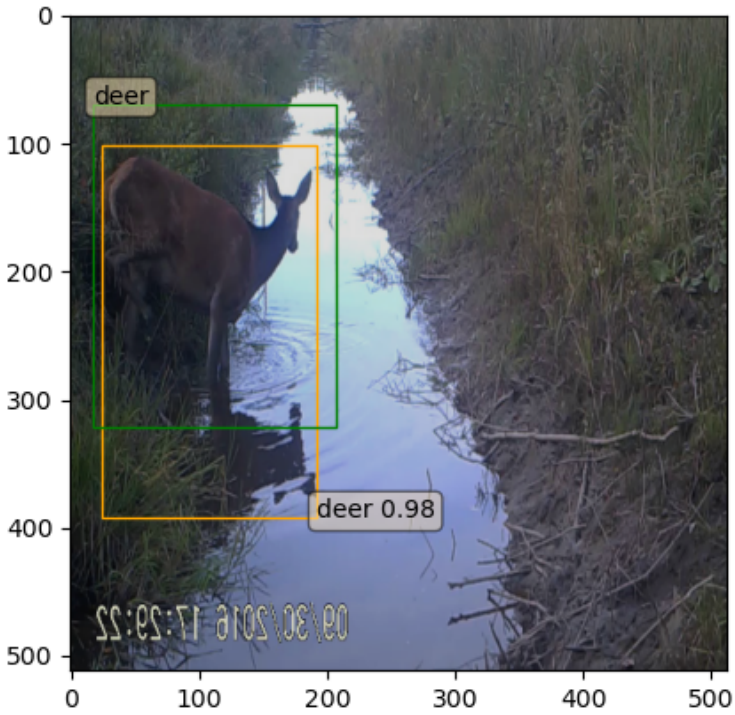

Figure 4.3: Faster R-CNN predictions after NMS is applied with threshold filtering and different species overlapping criterion. Image reflection is interpreted as the part of the animal.

overlapping species prediction (see Figure 4.2 and Figure 4.4).

4.2.1 One stage detectors

The most popular one stage detectors are SSD, YOLO and RetinaNet detectors [25], [26], [2]. YOLO predicts pre-defined anchor boxes for each cell with associated confidence of containing the object inside the bounding box and offset prediction. It also predicts the probability distribution of classes for each bounding box. Although YOLO produces many outputs, the confidence threshold filters out most of the bounding boxes. Recent YOLOv3 update YOLOv4 provided significant speed and accuracy improvement and introduced beneficial techniques such as anchor learning, mosaic augmentation, and Mish activation achieving state-of-the-art results of 65.7% mAP@0.5 for the COCO dataset [25]. YOLO is a fast model, which makes it preferable for real-time detection tasks, but it can struggle with small objects and be less accurate compared with Faster R-CNN.

Another one-stage detector RetinaNet stands out for its Focal Loss function, which is used to compensate the foreground-background class imbalance, which was assumed to be the significant problem that produced the one-stage and two-stage detector accuracy gap.

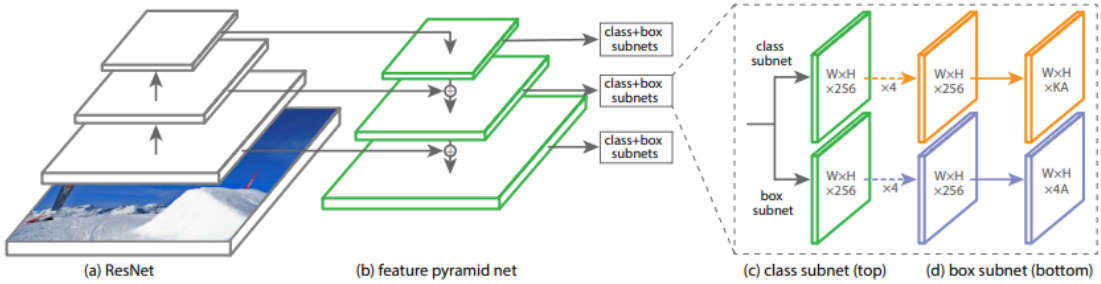

Figure 4.4: RetinaNet architecture. (a) Input image is passed through the ResNet backbone. (b) On top of it, FPN is applied, which combines up-sampling of the final backbone layer output with corresponding backbone layers. FPN enables to capture the high-level semantic features while not losing details. Each FPN layer has two heads for (d) regressing the anchor boxes to the ground truth bounding boxes and (c) classifying anchor boxes. [2]

4.2.2 Two stage detectors

The idea behind two-stage detectors is to propose regions where objects are potentially located and then iterate over them and perform classification and bounding box regression (bounding box coordinate offset prediction) by minimizing corresponding loss functions. Two-stage detector Faster R-CNN precursors (R-CNN, Fast R-CNN) relied on the unsupervised Selective Search algorithm for region proposal as shown in Figure 4.6. However, the Selective Search

Figure 4.5: Selective Search algorithm visualization. Adjacent pixels are clustered together based on the texture, color, intensity information [3].

algorithm is a fixed algorithm that bottle-necked the Fast R-CNN limiting the detector's speed.

This limitation was overcome with Faster R-CNN, which introduced RPN (Region Proposal Network) for generating regions that provided significant run-time improvement, which allowed Fast R-CNN to be used for the real-time detection [27].
In faster R-CNN model architecture (see Figure 4.6) the feature maps are obtained by running the input image through the convolution layers of the backbone network (e.g., ResNet50). Then region proposals are generated by RPN onto feature maps. Each region is reshaped with an ROI Pooling layer and passed to the model head that performs classification and regression. RPN learns to propose the regions by minimizing the objectness loss and regressing the proposed regions. The whole Faster R-CNN network has four losses, two losses for RPN learning and two losses for detector learning.

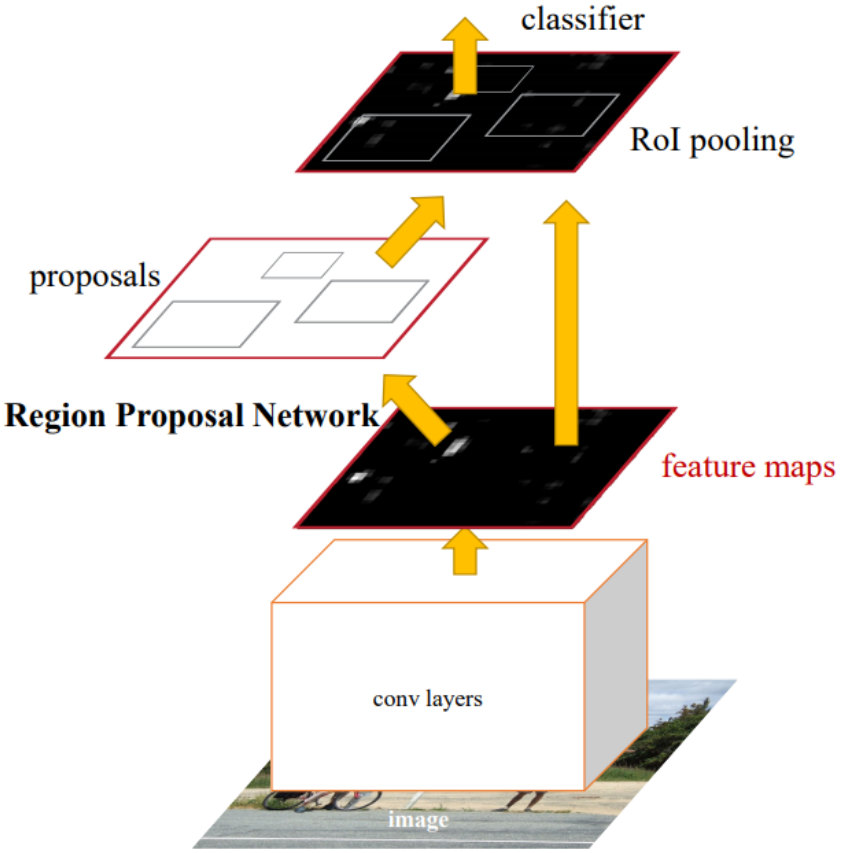

Figure 4.6: Faster R-CNN model architecture

5 Data and data pre-processing

5.1 Dataset

Dataset of interest was collected across the Latvian Jaunkalsnava and Kalsnava regions. Data consists of 8080 videos in 1280x720 resolution and 30FPS taken during four years. Videos were taken in the daylight and at night time using infrared lighting. Data examination reveals that many recordings are falsely triggered (do not contain animals), and remaining data mostly contain big mammals with a predominance of deer species. For this thesis, only wild boar (Sus scrofa) and dear (Cervidae family) representatives were chosen as the main species of interest. Thus smaller subset featuring mentioned species was selected (see Table 5.1).
Labeled image dataset was obtained by sampling annotated videos resulting in 880 annotated images (1042 annotations in total) featuring both day and nighttime captures and various object detection hazards, including motion blur, illumination variation, reflections, glare, and limited visibility of animals. Additionally, a dataset containing other species (including big bird species, goat, cat, dog, horse, fox, hare, human, mustelid, otter, raccoon and squirrel) was assembled.

Table 5.1: Test dataset

Species or group name	Scientific name	Number of annotations
Deer	*Cervidae*	516
Wild boar	*Sus Scrofa*	526
Other species		86
Total count		**1128**

Figure 5.1: Test dataset examples

Due to the small size of the obtained dataset, it was decided to use it for model testing. Further, in work, this dataset will be referred to as a test dataset. Some examples of images

included in this dataset are visualized in the Table 5.1.

5.2 Training data

In this work, animal datasets from the LILA BC repository were used to assemble large and balanced training dataset that will benefit model generality and prevent fast over-fitting. Data was gathered from the following datasets.

- Caltech Camera Traps (CCT) [28].
- Missouri Camera Traps [29].
- North America Camera Trap Images (NACTI) [14].
- WCS Camera Traps.
- Island Conservation Camera Traps.
- Channel Islands Camera Traps [30].
- ENA24-detection [31].
- Wellington Camera Traps [32].

Different datasets contain different levels of taxonomy, and it was decided to fuse animal species under Cervidae and Suidae families into ”deer” and ”boar” classes, respectively, resulting in the dataset with 9612 annotations in total. In order to balance the dataset, the oversampling approach was implemented, and 4234 additional boar images were additionally augmented during the training of the model, which resulted in a total of 13236 training dataset annotations (see Table 5.2).

Table 5.2: Training dataset

Species or group name	Scientific name	Number of annotations	Number of augmented samples	Total number of annotations
Deer	*Cervidae*	6970	0	6970
Wild boar	*Sus Scrofa*	2642	4328	6970
Total count		**9612**	**4328**	**13940**

5.3 Tools and Data format

Dataset consists of .jpg RGB digital images. Digital RGB image is a 3 dimensional matrix of pixel intensity values for red, green, blue channels respectively. Data also has annotations

stored in the .csv file containing ”id” annotation for file name, bounding box annotation in the ”xmin”, ”ymin”, ”xmax”, ”ymax” format and class id.

Pytorch and Torchvision libraries were used for the machine learning tool implementation, and Numpy, OpenCV, PIL, and Matplotlib libraries were used for the dataset arrangement scripts and visualization. Faster-RCNN and RetinaNet models were imported from the torchvision.models module with pre-trained weights on the COCO dataset. Model heads were replaced so they could predict only 2 classes (3 classes including background). The training was performed in the Google Colab Pro notebook environment with NVIDIA Tesla P100 GPU and tested on the PC.

5.4 Pre-processing and Training

5.4.1 Pre-processing

Images are initially re-scaled to 512x512 pixels size to lower the computational complexity of the convolution operations without losing much information of the object (comparing with the original size, it reduces convolution FLOPS $\sim$11 times [17]). On average, animals of interest occupy $\sim$6.23% of the image, which was enough to conserve the textures when down-scaled $\sim$3.52 times. Several initial image resolutions were tested, but 512x512 was selected as optimal in terms of speed and accuracy and assumed to be good enough to capture animal's distinctive features (see Figure 5.2).
After that, images are normalized by subtracting mean values from every pixel in R, G, B channels and dividing it with standard deviation values. Normalization reassures that model weights will not be biased to update in a particular direction and reassure that weights on the same layers will learn evenly, which will contribute to convergence properties of the network [33]. This happens because of the artificial neuron working principles, specifically linear combination and activation. Linear combinations of the unevenly scaled inputs will naturally favor specific inputs over another, and input distributions with a mean which is not close to the activation function mean will bias weight updates. These concepts are further expanded to the inner convolution layers as batch normalization, which normalizes previous layer outputs.

5.4.2 Data Augmentation

Following the image pre-processing step, data augmentation is applied. It is the technique that increases the number of data samples without gathering fundamentally new data. Augmentation can be implemented by adding modified copies of existing data or adding synthesized samples based on existing data. In this work, each image was flipped horizontally with 0.5 probability, and random brightness, contrast, and saturation variations in the non-extreme range

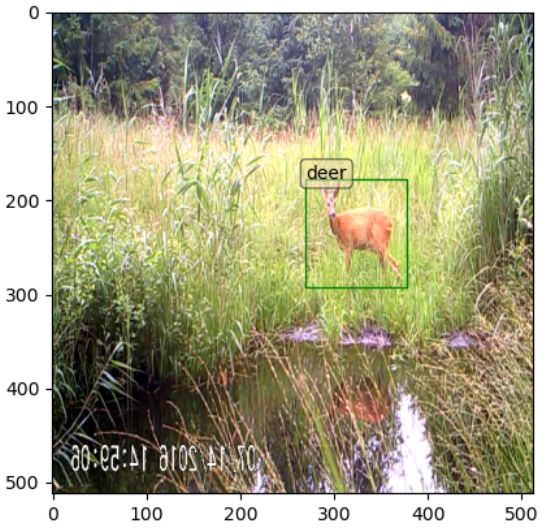

Figure 5.2: Example of the processed camera trap image with the visualised ground truth bounding box

were applied. These techniques simulate naturally occurring lighting variations, object pose transformations and thus contribute model robustness.

5.4.3 Transfer learning

When working with small dataset problems, one technique that can boost model performance is transfer learning, which refers to the reuse or fine-tuning of a particular model trained to accomplish similar tasks. State-of-the-art computer vision models frequently exploit pre-trained CNNs as a backbone network for the feature extraction, especially in case of working with a similar dataset, and build up the network over this model (e.g., EfficientPose) or fine-tune the pre-trained CNN to save time or compensate for a small training dataset while avoiding overfitting. Such model reuse is beneficial and allows to train models on smaller datasets.
Low-level features (i.e., learned first/second layer weights) for neural networks trained on the natural image datasets are highly similar and resemble Gabor filters, or color blobs [34]. They are almost task invariant, unlike the last layer features. Feature transfer with further fine-tuning can boost model performance even for relatively big datasets (645 000 images) [34]. Also, transferring features from similar dataset results in a smaller performance decline as we increase the number of transferred features compared with features from a dissimilar dataset, which means that it is possible to harness more layers without harming model performance [34]. In this thesis, several popular models will be implemented (ResNet50 and MobileNetV3) as the backbones with pre-trained weights on the COCO natural image dataset to investigate the significance of transfer learning for this task.

6 Results

6.1 Experiments

In order to assess the quantitative effect of different learning approaches and compare different models two experiments were designed:

1. 2 models: Faster RCNN-ResNet50 network and RetinaNet are trained for 34,850 iterations (10 epochs) on the training dataset with the following parameters:
   $batch\ size = 4,\ learning\ rate = 1e - 4$
   and Adam optimizer for the weight update. Other batch sizes and optimizers were tested, but those mentioned above were selected because they produced seemingly good convergence for the first training epoch. Both models were imported with pre-trained weights for the COCO image dataset.

2. To assess the effectiveness of the learning strategies, RetinaNet results from the first experiment are compared with the control cases featuring RetinaNet without pre-trained weights and RetinaNet results for the corresponding number of iterations on the not oversampled training set.

6.2 Metrics

Detection precision is measured in the mAP (Mean average precision) metrics commonly used to evaluate detection model performance. the mAP is calculated by averaging per-class mAP, which themselves are dependent on precision and recall inter-relation.
Precision represents the fraction of *good* bounding box predictions (IoU with ground truth bounding boxes $\geq$ IoU threshold) relatively to all predictions. The recall represents the fraction of *good* bounding box predictions relatively to all ground truth bounding boxes.

$$Precision = \frac{TP}{TP + FP} \tag{6.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{6.2}$$

Where:
TP = true positives.

FP = false positives

TN = true negatives.

FN = false negatives

$mAPc$ - per-class mean average precision

i - IoU threshold

k - number of classes

$$mAPc = \frac{1}{n} \sum_{i=1}^{n} Precision(Recall_i) \tag{6.3}$$

$$mAP = \frac{1}{k} \sum_{c=1}^{k} mAPc \tag{6.4}$$

6.3 Experiment 1

Faster R-CNN and RetinaNet were trained for ten epochs (34,850 iterations, 3485 iterations per epoch) on the training dataset with updating their weights each iteration. The results are represented in Table 6.1.

Table 6.1: Experiment 1: mAP evaluation

Model	**Metrics**	**Number of iterations**									
		3485	6970	10455	13940	17425	20910	24395	27880	31365	34850
Faster R-CNN	**mAP @0.5:0.05:0.95**	**0.1832**	**0.2697**	**0.2238**	**0.1913**	**0.1848**	**0.2618**	**0.2551**	**0.2449**	**0.2241**	**0.2582**
	mAP "deer"	0.1420	0.2584	0.2288	0.1062	0.1164	0.2336	0.1684	0.1877	0.1539	0.1576
	mAP "boar"	0.2244	0.2810	0.2187	0.2764	0.2532	0.2900	0.3417	0.3021	0.2942	0.3589
	mAP@0.5	0.3229	0.4561	0.3934	0.3305	0.3148	0.4562	0.4073	0.4065	0.3776	0.4204
	mAP "deer"	**0.2800**	**0.4737**	**0.4337**	**0.2154**	**0.2098**	**0.4332**	**0.3065**	**0.3414**	**0.2956**	**0.2996**
	mAP "boar"	**0.3657**	**0.4385**	**0.3531**	**0.4456**	**0.4197**	**0.4791**	**0.5080**	**0.4715**	**0.4596**	**0.5411**
	mAP @0.75	0.1932	0.2860	0.2229	0.1926	0.1959	0.2855	0.2758	0.2571	0.2488	0.2756
	mAP "deer"	**0.1367**	**0.2671**	**0.2222**	**0.0970**	**0.1175**	**0.2218**	**0.1659**	**0.1881**	**0.1454**	**0.1536**
	mAP "boar"	0.2496	0.3048	0.2235	0.2881	0.2743	0.3492	0.3857	0.3260	0.3521	0.3976
RetinaNet	**mAP @0.5:0.05:0.95**	**0.2158**	**0.2016**	**0.2413**	**0.2046**	**0.2346**	**0.2494**	**0.2659**	**0.2364**	**0.2202**	**0.2192**
	mAP "deer"	0.1287	0.1884	0.1715	0.1791	0.1827	0.1757	0.2053	0.2098	0.1551	0.1844
	mAP "boar"	0.3029	0.2148	0.3111	0.2301	0.2865	0.3231	0.3266	0.2630	0.2853	0.2540
	mAP@0.5	0.3740	0.3725	0.4133	0.3574	0.4134	0.4198	0.4364	0.4173	0.3738	0.3922
	mAP "deer"	**0.2727**	**0.3776**	**0.3361**	**0.3473**	**0.3530**	**0.3437**	**0.3814**	**0.4021**	**0.3017**	**0.3789**
	mAP "boar"	**0.4752**	**0.3673**	**0.4904**	**0.3675**	**0.4737**	**0.4959**	**0.4913**	**0.4325**	**0.4458**	**0.4054**
	mAP @0.75	0.1996	0.1909	0.2483	0.2179	0.2666	0.2678	0.2890	0.2421	0.2341	0.2236
	mAP "deer"	**0.1028**	**0.1473**	**0.1499**	**0.1642**	**0.1844**	**0.1631**	**0.2152**	**0.1929**	**0.1442**	**0.1556**
	mAP "boar"	0.2963	0.2345	0.3467	0.2716	0.3487	0.3724	0.3628	0.2913	0.3240	0.2915

mAP metric is used for the model evaluation. mAP for 0.5 and 0.75 IoU threshold values are measured and averaged mAP for IoU threshold values between 0.5 and 0.95 with a step size of 0.05.

The best mAP@0.5:0.05:0.95 of **0.2786** was obtained by the Faster R-CNN-ResNet50 model already on the second epoch. However, after that, Faster R-CNN mAP drastically decreases (almost by 0.1) during the subsequent three epochs and then slightly recovers, reaching 0.2582 in the 10th epoch. The best mAP@0.5:0.05:0.95 score for the RetinaNet is **0.2659** which was obtained on the 7th epoch and is comparably close to the Faster-RCNN best result. Unlike the Faster R-CNN, RetinaNet demonstrated more stable precision dynamics. After eight epochs,

both model precision does not improve explicitly. However, more certain conclusions require further testing of the model.

Per class mAP dynamics for the RetinaNet (see Figure 6.3) shows that model was able to

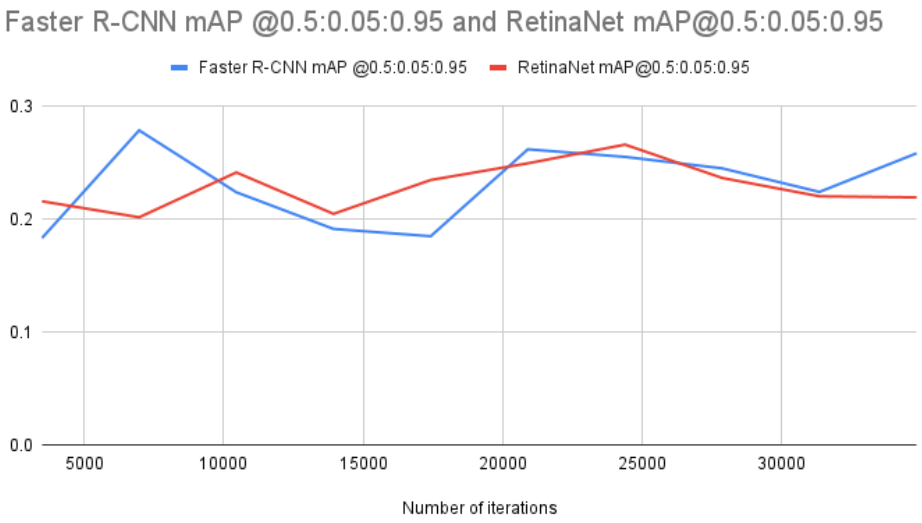

Figure 6.1: mAP@0.5:0.05:0.95 score dynamics

learn to detect boars more successfully, which was the main initial concern. A similar pattern is observed for the Faster R-CNN (see Figure 6.3) with higher detection precision (by $\sim$0.1 on average) of boars.

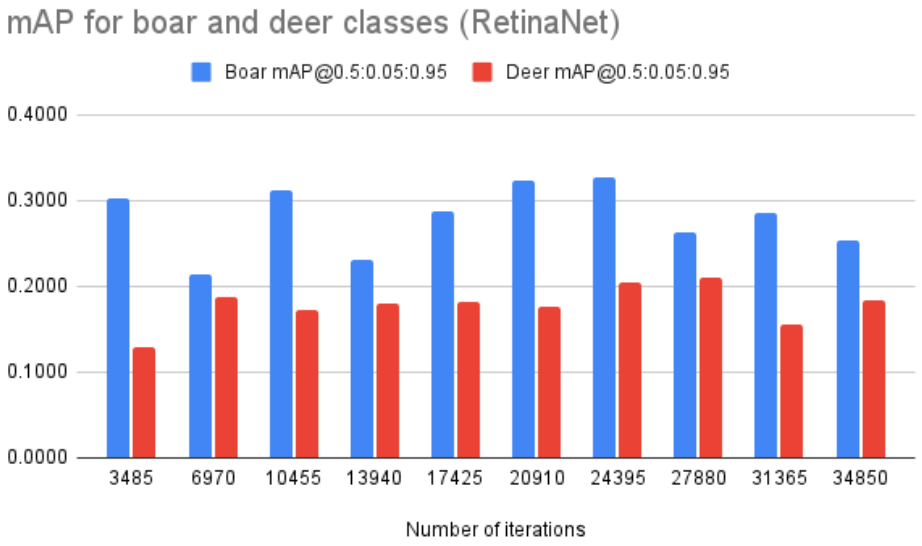

Figure 6.2: RetinaNet mAP@0.5:0.05:0.95 for ”boar” and ”deer” classes

The best performing models were also tested on the ”other species” datasets containing captures featuring various other animals to evaluate model robustness. The Faster R-CNN trained for the two epochs and RetinaNet trained for seven epochs were selected. Faster R-CNN predicted 0 detection predictions out of potential 78, whereas RetinaNet predicted 10 detections.

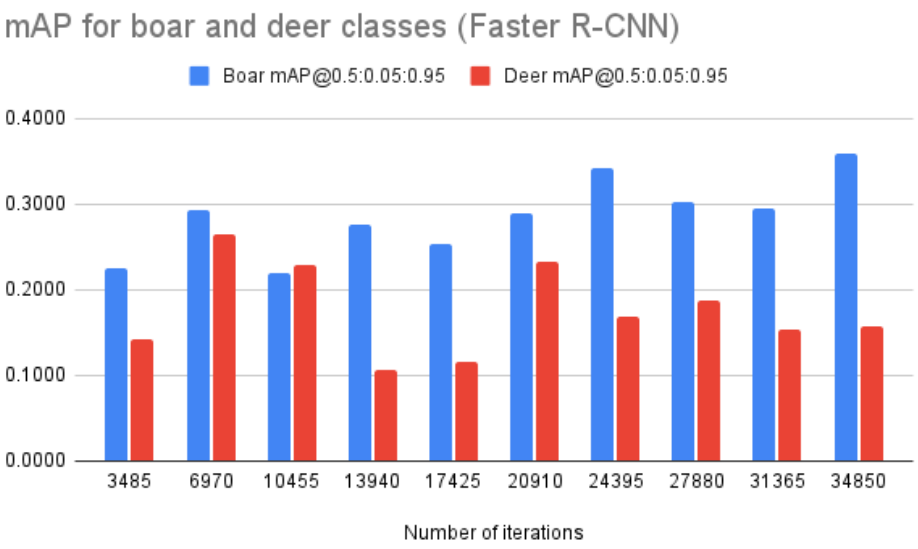

Figure 6.3: Faster R-CNN mAP@0.5:0.05:0.95 for ”boar” and ”deer” classes

Most of these predictions mistaken background for animals and had low probability, suggesting that a higher probability threshold can exclude most of them. Overall, despite not having the *negative* class with all other animals during training, the models were able to learn very distinctive features of the boars and deer that were enough not to confuse them with other animals.

6.3.1 Run-time and memory analysis

Although fast run-time is not a very important criterion for the specificity of this work as the model is not intended to run in real-time, it is still the benefit and could pose potential usage in the form of the computational block on the camera itself or some other real time solution such as collision avoidance system module. Results are summarised in the Table 6.2.

Table 6.2: Run-time

Metrics	Model	
	Faster R-CNN ResNet50	RetinaNet ResNet50
prediction time (s)	0.3132	0.2838
processing time (s)	0.001	0.0043
total time (s)	0.3142	0.2881

Execution times are averaged for the whole test set (1042 images) and are tested on the PC (CPU: Intel(R) Core(TM) i7-9700K @ 3.60GHz, GPU: Nvidia GeForce GTX1660S). Processing time shows the average time for performing NMS and transforming the data. RetinaNet appears to be slightly faster surpassing the Faster R-CNN by 0.0261 seconds, which could be related to the one-stage architecture of the RetinaNet, which jointly regress the bounding boxes and classification, which is generally faster than two-stage methods. However, the processing

time is four times larger than Faster R-CNN, which is related to the generation of more predictions. The expected FPS is 3.18FPS and 3.47FPS for Faster R-CNN and RetinaNet, which is quite low for real-time implementation. Faster R-CNN takes 965.5 MB on the disk, and RetinaNet takes 752 MB of disk space, decreasing the mobility of these models.

6.3.2 Loss convergence

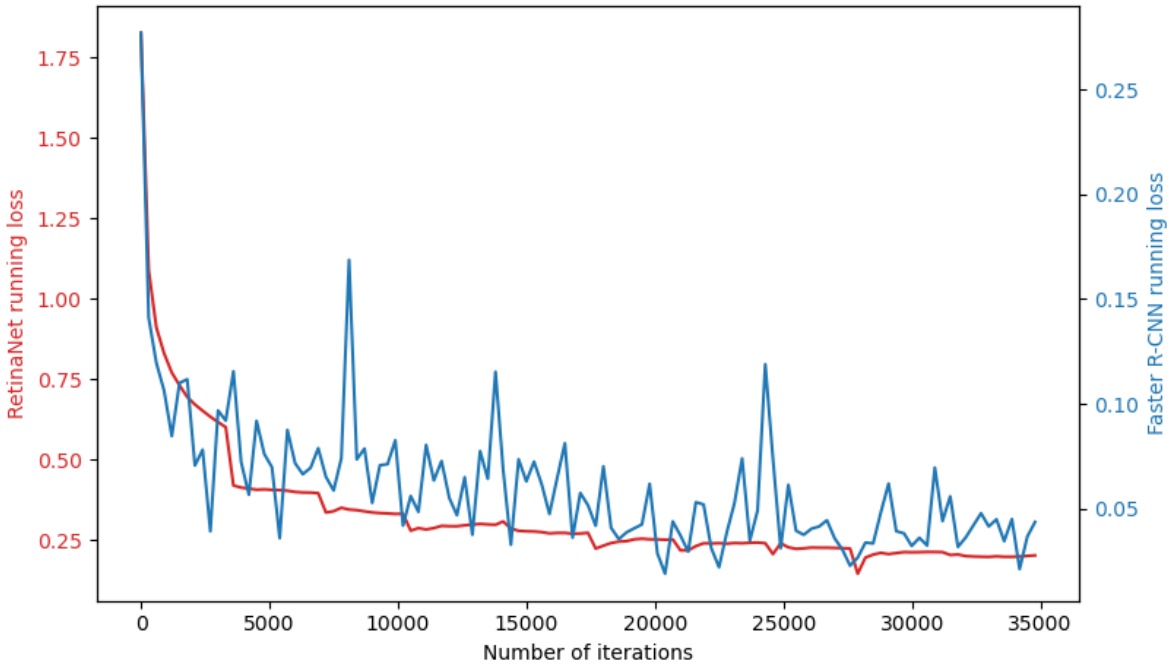

Figure 6.4: Losses sum convergence for Faster R-CNN and Retina Net. Red line shows the Retina Net loss sum and blue line shows the Faster R-CNN loss sum.

On the Table 6.4 the loss sum is plotted for RetinaNet and Faster R-CNN for every 300th iteration. The RetinaNet loss converges exponentially, whereas Faster-RCNN loss fluctuates, and its overall downward trend is more ambiguous, indicating that the model is not learning efficiently. Even though Faster R-CNN loss function is often noisy due to the RPN learning (e.g., [35]), obtained fluctuations have a high amplitude, and the possible reason could be inefficient learning hyper-parameters.

At the beginning of each epoch, there are sharp changes in the RetinaNet loss caused by the learning rate update with Adam Optimiser. In order to avoid this, a more frequent learning rate update with decay could be implemented.

6.4 Experiment 2

To assure that learning optimization techniques have been effective, 2 additional negative control cases were tested: RetinaNet without pre-trained weights, pre-trained RetinaNet with not over-sampled data (see Table 6.3)

Table 6.3: Experiment 2: mAP evaluation

Metrics	**Model**		
	RetinaNet pre-trained	RetinaNet not pre-trained	RetinaNet pre-trained (not oversampled dataset)
mAP @0.5:0.05:0.95	0.2158	0.1695	0.2290
mAP ”deer”	0.1287	0.1492	0.1953
mAP ”boar”	0.3029	0.1897	0.2626
mAP@0.5	0.3740	0.2989	0.4029
mAP ”deer”	0.2727	0.2900	0.3758
mAP ”boar”	0.4752	0.3078	0.4299
mAP @0.75	0.1996	0.1688	0.2265
mAP ”deer”	0.1028	0.1441	0.1714
mAP ”boar”	0.2963	0.1935	0.2815

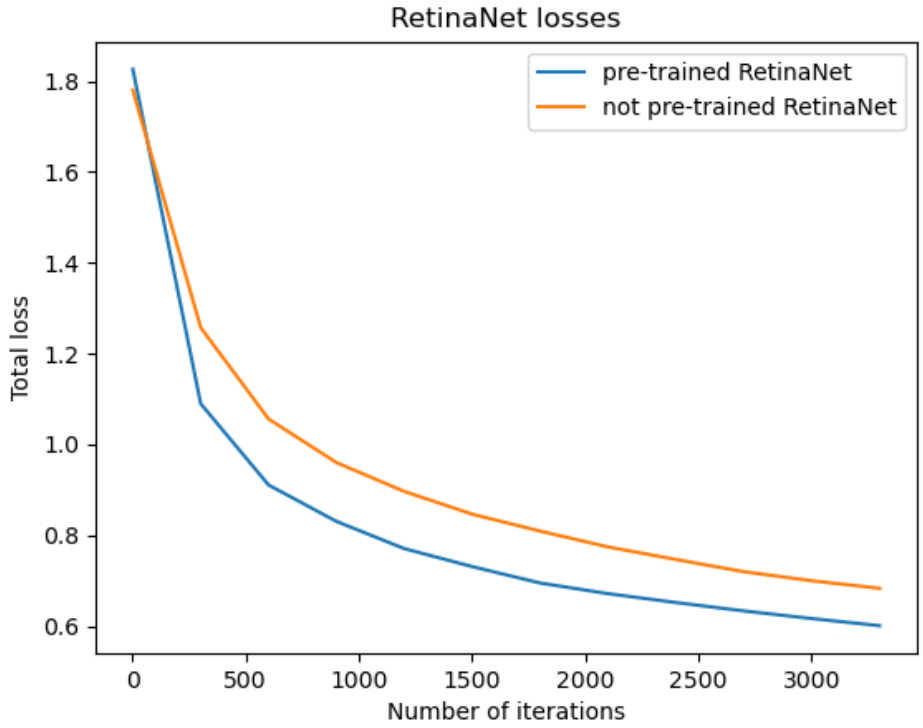

Figure 6.5: Loss functions for pre-trained and not pre-trained RetinaNet

Without pre-trained weights RetinaNet is performing worse when trained for one epoch. From the Table 6.5 it seems that loss values for not pre-trained network converge slower. Also

mAP@0.5 score is lower by 0.0751 and mAP@0.5:0.05:0.95 is lower by 0.0463. We can conclude that pre-trained network weights have successfully increased learning efficiency of the network.

RetinaNet trained on the not oversampled dataset achieved larger per-class mAP score differences compared with pre-trained RetinaNet trained on the oversampled dataset which contradicts with the intention of the oversampling implementation in this work. For the mAP@0.5:0.05:0.95 (see Table 6.3) difference is 0.1742 compared with 0.0673 ($\sim$ 2.5 times larger). This can be temporal effect which will gradually dissolve, though it could also be the case where oversampling deteriorate the model precision. From the Table 6.3 we see that oversampling improved "boar" class accuracy, but drastically decreased "deer" class accuracy.

Faster R-CNN trained on the not over-sampled data has uneven mAP scores for "deer" and "boar" compared to first experiment results. Faster-RCNN from the first experiment show almost even mAP for the "deer" and "boar" classes which suggests that the model was able to learn in unbiased manner and rare case oversampling was successful.

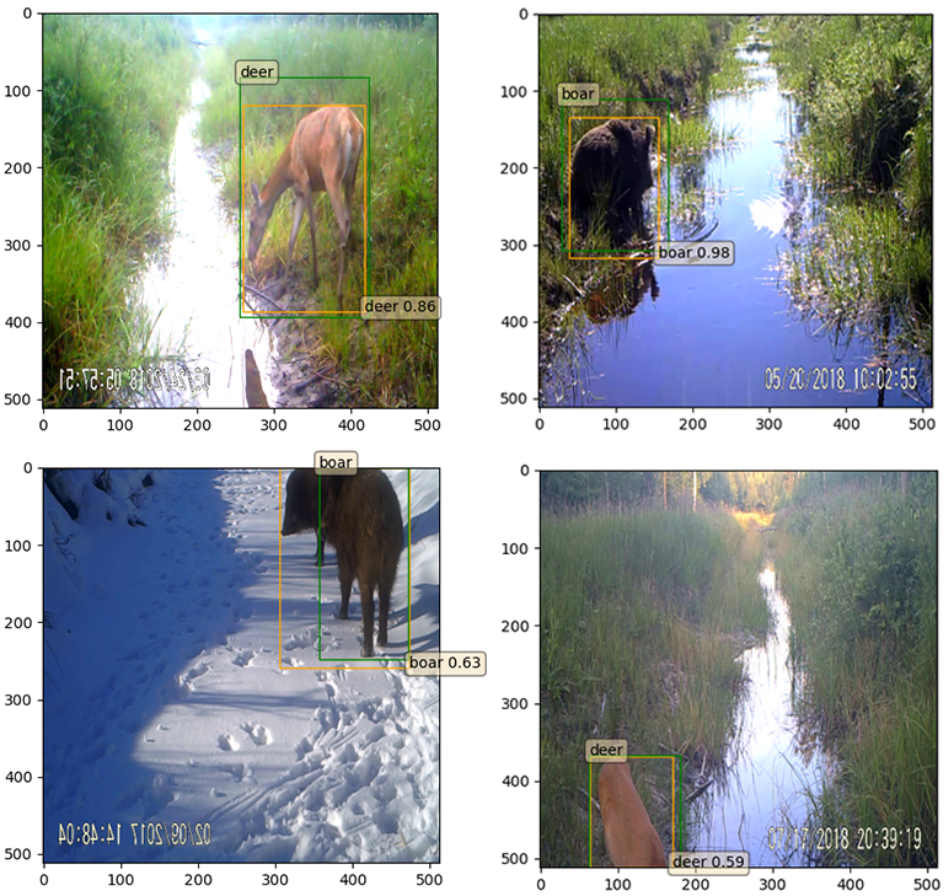

Figure 6.6: Prediction examples.

7 Discussion

mAP evaluation and loss function convergence suggest that both models have successfully learned to detect ”boar” and ”deer” with average precision exceeding 25%. After the eighth epoch, the performance of both models starts to decrease, which could be interpreted as over-fitting. Network over-fitting refers to the detrimentally high test dataset feature approximation, which lowers the model generality. However, it is hard to determine the real reason because the models were not tested for more epochs, and such a decrease could be temporal.
mAP differences for the ”deer” and ”moose” classes can be related to the certain biases in the training and test set, and relatively small test set size, which means that individual detection complications have a more significant impact on the precision result.
With the assumption that the problem occurs on the classification and not on the localization stage, deer family species variability could be a potential cause of miss-classifications. Various deer species presented in the dataset could visually look very different, such as roe deer, moose, and red deer. From the prediction examination, it was observed that moose is more frequently miss-classified with boars in Figure 7.1. Thus, species-specific classification will force the model to develop individual weights for the ”moose” class, leading to better classification accuracy.
From the other perspective, detection differences can be caused by failed detections of deers.

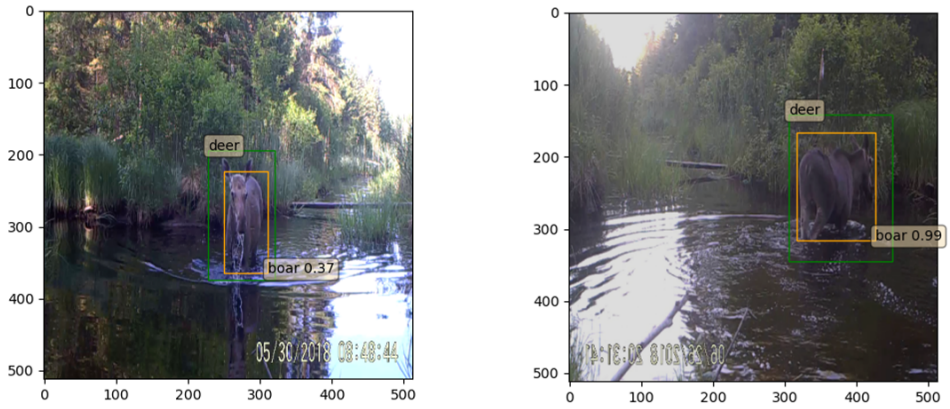

Figure 7.1: Detection examples where moose is miss-classified as boar. Also, ground truth bounding box inaccuracies are seen on these images.

Deer are more frequently captured on cameras facing large open areas. As a result, the deer are frequently captured on the big distances, making deer look smaller, which provides less infor-

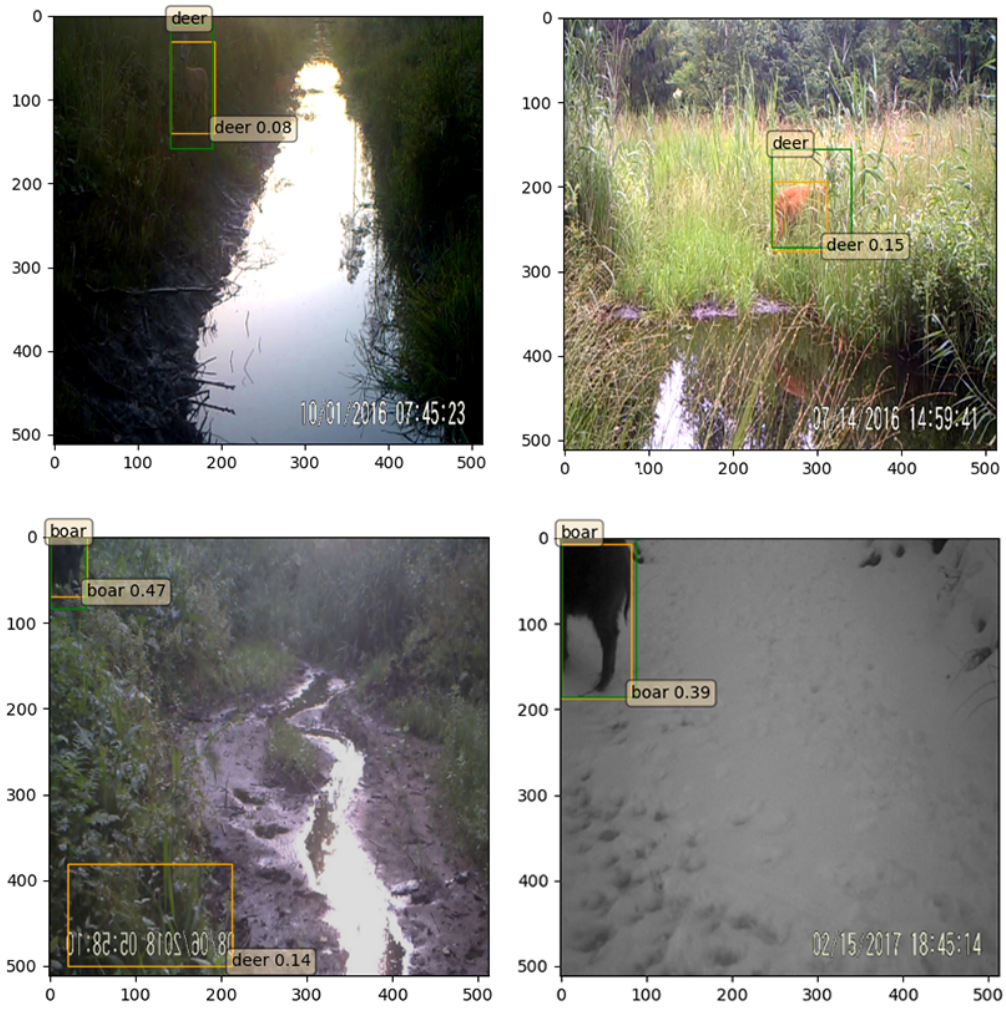

Figure 7.2: Examples of detection predictions for challenging captures.

mation for successful localization and classification. It is a severe challenge for Faster R-CNN because its RPN has a fixed receptive field and handles discrepancy of object sizes multi-path architecture solutions are required. Also, deer are more frequently captured in or near ponds, so these captures usually have clear deer reflection, which is sometimes included in the bounding box, resulting in smaller IoU between the grounding box and predicted box. From the other perspective, boars are more frequently captured on the close shots, which focuses on the smaller region and thus more frequently produces images where boars are cropped out, which also limits high precision detection.

Overall, despite the not very precise ground truth annotations for the test set (shown in Figure 7.2) which lowered mAP for high IoU thresholds and other complications such as animal reflections and limited visibility, the model were able to handle quite complicated detection cases. In order to obtain more representative scores, more captures should be annotated, and models should be trained for more epochs on the larger dataset.

8 Conclusions and Future Work

In this work, Faster R-CNN and RetinaNet deep neural networks were successfully trained to detect animals on the camera trap images and obtained maximal mAP@0.5 of 0.4562 and 0.4364 and mAP@0.5:0.05:0.95 of 0.2786 and 0.2659, respectively. Comparison with the Faster R-CNN benchmark on the COCO test-dev (mAP@0.5 65.7%) suggests that model precision is relatively good, considering detection hazard abundance in the test dataset. mAP@0.5 of 0.4562 is good enough to alleviate annotation work to such model partly. Additionally, an animal dataset assembling script was developed during this work, which gathers species of interest from the most extensive animal camera trap datasets, and learning optimization techniques were implemented and evaluated.

The current model has many limitations and drawbacks which could be addressed and improved in future works. Obtained networks can detect only two classes: ”deer” and ”boar”. However, it is possible to train the model to distinguish between deer species such as ”roe deer”, ”red deer” and ”moose”, which will produce more valuable biological information.

From the learning optimization perspective, more advanced data augmentation techniques such as the generation of synthetic data (e.g., simulated captures produced by the 3D engine) could be used to amplify the number of training data while avoiding data similarity. Also, as shown in this work, initial assumptions about rare cases could not result in the desired balanced detection precision. This problem can be potentially overcome by performing emphasis learning [11] which will focus on miss-classified samples which not necessarily belong to the rare class. Also, hyperparameter tuning could be implemented to find balanced parameters. Video capture advantages were not harnessed in this work, so this opportunity is still available to be implemented in future works.

References

[1] Comparison between image classification, object detection and instance segmentation. https://miro.medium.com/max/3000/1*hz6t-tokg1niaufmcysusw.jpeg.

[2] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection, 2018.

[3] Jasper Uijlings, K. Sande, T. Gevers, and A.W.M. Smeulders. Selective search for object recognition. International Journal of Computer Vision, 104:154–171, 09 2013.

[4] Ana M. Valente, Pelayo Acevedo, Ana M. Figueiredo, Carlos Fonseca, and Rita T. Torres. Overabundant wild ungulate populations in europe: management with consideration of socio-ecological consequences. Mammal Review, 50(4):353–366, 2020.

[5] Antonio J. Carpio, Marco Apollonio, and Pelayo Acevedo. Wild ungulate overabundance in europe: contexts, causes, monitoring and management recommendations. Mammal Review, 51(1):95–108, 2021.

[6] Jochen Langbein, Rory Putman, and Bostjan Pokorny. Traffic collisions involving deer and other ungulates in Europe and available measures for mitigation, page 215–259. Cambridge University Press, 2011.

[7] Franck Trolliet, Marie-Claude Huynen, Cédric Vermeulen, and Alain Hambuckers. Use of camera traps for wildlife studies. a review. Biology Agriculture Science Environnement, 18:446–454, 01 2014.

[8] AB Swanson, M Kosmala, CJ Lintott, RJ Simpson, A Smith, and C Packer. Data from: Snapshot serengeti, high-frequency annotated camera trap images of 40 mammalian species in an african savanna, 2015.

[9] Zhongqi Miao, Kaitlyn Gaynor, Jiayun Wang, Ziwei Liu, Oliver Muellerklein, Mohammad Sadegh Norouzzadeh, Alex Mcinturff, Rauri Bowie, Ran Nathan, Stella Yu, and Wayne Getz. Insights and approaches using deep learning to classify wildlife. Scientific Reports, 9, 05 2019.

[10] Mohammed Sadegh Norouzzadeh, Anh Nguyen, Margaret Kosmala, Ali Swanson, Meredith Palmer, Craig Packer, and Jeff Clune. Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning, 2017.

[11] H. He and E. A. Garcia. Learning from imbalanced data. IEEE Transactions on Knowledge and Data Engineering, 21(9):1263–1284, 2009.

[12] Christin Carl, Fiona Schönfeld, Ingolf Profft, Alisa Klamm, and Dirk Landgraf. Automated detection of european wild mammal species in camera trap images with an existing and pre-trained computer vision model. European Journal of Wildlife Research, 66, 07 2020.

[13] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, and et al. The open images dataset v4. International Journal of Computer Vision, 128(7):1956–1981, Mar 2020.

[14] Michael A. Tabak, Mohammad S. Norouzzadeh, David W. Wolfson, Steven J. Sweeney, Kurt C. Vercauteren, Nathan P. Snow, Joseph M. Halseth, Paul A. Di Salvo, Jesse S. Lewis, Michael D. White, Ben Teton, James C. Beasley, Peter E. Schlichting, Raoul K. Boughton, Bethany Wight, Eric S. Newkirk, Jacob S. Ivan, Eric A. Odell, Ryan K. Brook, Paul M. Lukacs, Anna K. Moeller, Elizabeth G. Mandeville, Jeff Clune, and Ryan S. Miller. Machine learning to classify animal species in camera trap images: Applications in ecology. Methods in Ecology and Evolution, 10(4):585–590, 2019.

[15] Mohammad Sadegh Norouzzadeh, Dan Morris, Sara Beery, Neel Joshi, Nebojsa Jojic, and Jeff Clune. A deep active learning system for species identification and counting in camera trap images, 2019.

[16] 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, 18(10):2079–2092, 2016.

[17] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2020.

[18] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning. MIT Press, 2016. `http://www.deeplearningbook.org`.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 770–778, 2016.

[20] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[21] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. CoRR, abs/1602.07261, 2016.

[22] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger, 2016.

[23] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection, 2017.

[24] Christopher Michael Bishop. Pattern Recognition and Machine Learning. Springer Science+Business Media, 2006.

[25] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection, 2020.

[26] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. Lecture Notes in Computer Science, page 21–37, 2016.

[27] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. CoRR, abs/1506.01497, 2015.

[28] Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors. Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XVI, volume 11220 of Lecture Notes in Computer Science. Springer, 2018.

[29] Zhi Zhang, Zhihai He, Guitao Cao, and Wenming Cao. Animal detection from highly cluttered natural scenes using spatiotemporal object region proposals and patch verification. IEEE Transactions on Multimedia, 18(10):2079–2092, 2016.

[30] The nature conservancy (2021): Channel islands camera traps 1.0. the nature conservancy. dataset.

[31] Hayder Yousif, Roland Kays, and Zhihai He. Dynamic programming selection of object proposals for sequence-level animal species classification in the wild. IEEE Transactions on Circuits and Systems for Video Technology, 2019.

[32] Victor Anton, Stephen Hartley, Andre Geldenhuis, and Heiko U Wittmer. Monitoring the mammalian fauna of urban areas using remote cameras and citizen science, 2018.

[33] Yann A. LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Efficient BackProp, pages 9–48. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[34] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks?, 2014.

[35] Wang, Chen Guanhua, Cao Yun, An Feng, Xue, and Ting Yun. Individual rubber tree segmentation based on ground-based lidar data and faster r-cnn of deep learning. Forests, 10:793, 09 2019.

Non-exclusive license to reproduce thesis and make thesis public

I, Ilja Pavlovs,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Animal recognition using deep learning

   supervised by Prof. Gholamreza Anbarjafari and Egils Avots.

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Ilja Pavlovs **20.05.2021**