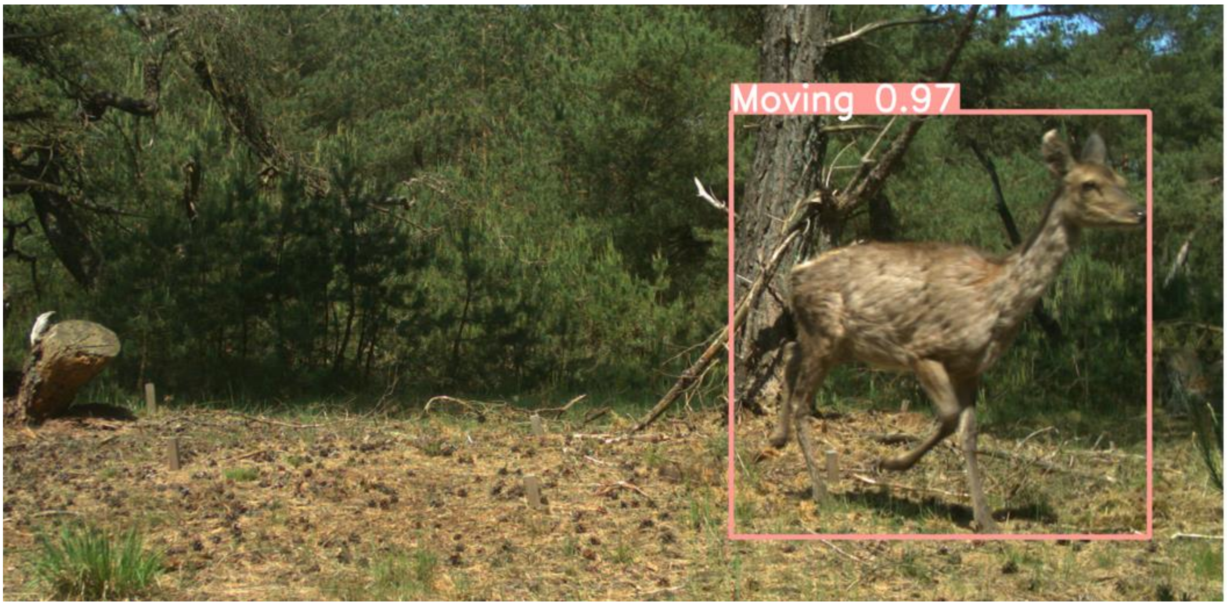
Wildlife Ecology and Conservation Group

MSc Thesis Wildlife Ecology and Conservation

Recognition of wildlife behaviour in camera-trap photographs using machine learning

Jorrit van Gils

01/04/2022



WAGENINGEN
UNIVERSITY & RESEARCH

Action recognition of wildlife in camera-trap photographs using machine learning

A comparison of YOLOv5 and the pose estimation approach

Name course	: Thesis Wildlife Ecology and Conservation
Number	: WEC-80436
Date	: 21/02/2022

Student	: JN (Jorrit) van Gils
Registration number	: 1006511
Study programme	: MSc Forest and Nature Conservation

Supervisors	: Dr. ir PA (Patrick) Jansen
	Dr. ir HJ (Henjo) de Knegt
External advisors	: H (Helena) Russello (Farm Technology)
	Dr. GW (Gert) Kootstra (Farm Technology)
	Drs. R. Hollands (Verrassend Hollands)

Group	: Wildlife Ecology and Conservation Group
	6708 PB Wageningen
	T: +31 (317) 48 58 28
	E: patricia.meijer@wur.nl

WAGENINGEN
UNIVERSITY & RESEARCH

Disclaimer

This report is written by a master student Forest and Nature Conservation guided by the Wildlife Ecology and Conservation chair group. Using the information in this thesis is at own risk. It is highly recommended to check the information before it is applied. The University of Wageningen will not be responsible by law, for the consequences of this report. Reproduce or publish information from this report is only allowed with explicitly approval of:

Wageningen University and Research

Wildlife Ecology and Conservation Group

6708 PB Wageningen

T: +31 (317) 48 58 28

E: patricia.meijer@wur.nl

Preface

Before you lies the Msc thesis ‘Action recognition of wildlife in camera-trap photographs using machine learning’. I would like to thank my supervisors dr. ir. Patrick Jansen and dr. ir. Henjo. de Knegt for their excellent guidance and support.

The research was challenging because the subject, largely about deep learning was not included as standard in the Forest and Nature Conservation curriculum. Fortunately, my external supervisors H. Russello, Dr. G.W. Kootstra and Drs. R. Hollands were always available and willing to answer my deep learning related queries.

Finally, I would like to express gratitude to, J.E. Doornweerd, D. Marcos Gonzalez, B. Jackson, R. Bloo, N. Heida, L. Dijkhuis, B. Sluiter, S. Beery, A. Maxwell and J. Feyen for the wonderful cooperation as well.

Abstract

With more than a million cameras to monitor wildlife actions, human analysis of camera-trap data has become time-consuming and unsustainable (Tuia et al., 2022). Deep learning algorithms that can automatically extracts features from photographs, provides a powerful solution (Schneider, Taylor, Linquist, & Kremer, 2019) but are not widely used yet for automated wildlife action recognition (W. Li, Swetha, & Shah, 2020).

Using deep learning for wildlife presented the added challenge of a highly varying background, causing models that classified objects based on full images to often predict well on training data but poorly on new test data (Beery, Van Horn, & Perona, 2018). As new images still contain the same species, but in a different background, models that classify based on the entire photo seem to experience distraction from the background. Deep learning object detection methods have come up with a clever way to overcome this problem by classifying based on the localization of the animal, reducing background distortion. Two object detection methods have grown in popularity especially in recent years, YOLOv5 and Pose estimation.

In a case study on Red deer (*Cervus elaphus*) in National Park Hoge Veluwe in The Netherlands YOLOv5 and Pose Estimation were applied to examine their predictive capacity to new data. A relatively small dataset of 506 single Red deer (*Cervus elaphus*) images labelled with actions 'foraging', 'moving' or 'other' was split into an independent train and test dataset. YOLOv5 classified the action of an animal based on a rectangle bounding box fitted around the animal. In the pose estimation approach (PEA), the action was classified by a random forest algorithm that used key-point features as input extracted from pose estimation technique DeepLabCut (DLC). Both methods were evaluated by accuracy, prediction, recall and human-effort. During comparison methodological differences were considered.

It was hypothesized that PEA compared to YOLOv5 would reach higher performance as for pose estimation less training data that was needed (Nath et al., 2019) and it had fine-grained control by only looking at the relevant properties of an object (Mathis, Schneider, Lauer, & Mathis, 2020). Contrary to expectations, YOLOv5 outperformed PEA by achieving a higher accuracy (0.55) compared to PEA (0.53), being six times less time consuming, requiring less computational power and by user-friendly inference in which you apply the model to new data. Predictive capacity of YOLOv5 is mainly expected to improve by increasing the datasets class sizes (Shahinfar, Meek, & Falzon, 2020). Although results point in favour of the YOLOv5 model, there is a suspicion that PEA does have a lot of potential, but in hindsight the sub-optimal set-up of PEA may have led to a lower predictive capacity. It should be further investigated to what extent adapting the labelling strategy and using bounding boxes as input for PEA can contribute to levelling or perhaps even improving YOLOv5. While these individual action recognition models are already a great step for ecology, for a more extensive applicability it is advised to train the models with pictures containing multiple Red deer (*Cervus elaphus*) individuals and by classifying behaviour based on animal's action changes of successive images.

Table of contents

1. Introduction 7
1.1 Introduction 7
1.2 This study 9
1.3 Outline........ 9
2. Methods........ 10
2.1 Study system 10
2.2 Taxonomy of actions........ 10
2.3 Dataset creation........ 12
2.4 Object detection 14
2.4.1 YOLOv5........ 15
2.4.2.1 Box 1. Origin and how YOLOV5 works 17
2.4.2 Pose estimation approach 18
2.4.2.1 DeepLabCut........ 18
2.4.2.2 Random forest 19
2.4.2.3 Box 2. The history of pose estimation 21
2.4.3 Background 22
2.4.3.2 Box 3. The history of deep learning 22
2.4.3.2 Box 4. Automated action recognition of wildlife 24
3. Results........ 25
4. Discussion........ 28
5. Reference list 31
Appendix 1: Map National Park Hoge Veluwe........ 35
Appendix 2: Action annotation protocol 36
Appendix 3: Scripts and models........ 37
Appendix 4: BOX21 example input 38
Appendix 5: Train and test error graphs........ 39
Appendix 6: DLC annotation protocol........ 40
Appendix 7: PEA key-point features and distribution........ 42
Appendix 8: Results........ 44

1. Introduction

1.1 Introduction

Studying the behaviour of wild animals, is a topic of great importance to ecologists (Braude, Margulis, & Broder, 2017). Knowing the behaviour helps to understand the animals' requirements, its internal state, needs, preferences and dislikes (Mench, 1998). Providing ecologists with accurate and large-scale knowledge about the animal behaviour can help to inform ecology and management (Norouzzadeh et al., 2018). Recognizing animals actions is the foundation for understanding their behaviour (Lehner, 1992).

Over the past decades techniques to recognize wildlife actions have changed rapidly from using ethograms (Altmann, 1974) to the use of animal tags (Prokhorov, 1970). Constrain to both of these methods were that the observer has to be in the vicinity of the animal before or during the observation (Schneider et al., 2019). Tagging was considered laborious, expensive (Schneider et al., 2019), harmful (Mellor, Beausoleil, & Stafford, 2004) and displacement data did not even show the action an animal was doing. Most importantly, tagging was unable to scale to large populations (X. Li, Cai, Zhang, Ju, & He, 2019).

Camera-traps are an alternative way to measure actions by recording images. Camera-traps provide a window into the animals world, with lower costs and with a reduced workload for researchers (Braude et al., 2017; Norouzzadeh et al., 2018; Schneider et al., 2019) and therefore are among the most used sensors by ecologists (Tuia et al., 2022). Especially when classification of actions can be automated, the use of camera-traps can have a lot of potential (Pereira et al., 2019; Schneider et al., 2019).

The standardized and automated alternative to manual image analysis is computer vision (Schneider et al., 2019), in which computers are trained to interpret the visual world (A. Zhang, Lipton, Li, & Smola, 2020). At the start of computer visions part-based models were used in which models could only recognize parts of an object separately and manual calculations of relative distances between these parts were required to make a binary choice whether something was an object or not (Fergus, Perona, & Zisserman, 2003).

In recent years computer vision has evolved rapidly due to deep learning, which is a subdivision of machine learning that deals with providing computers the ability to learn, without being explicitly programmed (Hastie, Tibshirani, Friedman, & Friedman, 2009). Training a machine learning model requires all data to be labelled. During training the labelled training data is used by the model to learn to make correct predictions, and the model is tested with the testing data with hidden labels. To find the accuracy of the model, these hidden labels are retrieved and compared with the models predictions (A. Zhang et al., 2020). Deep learning evolved from machine learning by having multi-layered neural networks, that attempted to simulate the action of the human brain. These DL-models turned out to be extremely powerful because according to the universal approximation theorem only a single layer network can already make an approximate prediction (Kratsios, 2021).

Because deep learning algorithms included automated feature extraction it was not needed anymore to calculate features like the distance between the muzzle and the toe manually, which improved objectivity and removed human bias in which researchers assumed what features are important (Schneider et al., 2019). Automated feature extraction could be imagined by the network recognizing the presence of small- or large objects, the position of objects in the image or the colour contrast. Although automatic feature extraction was at the expense of interpretability and deep learning models were often referred to as a black box (Waldrop, 2019), deep learning algorithms often outcompeted traditional machine learning methods especially when using large datasets (Freytag et al., 2016; Schneider et al., 2019).

Despite its potential, deep learning applied to wildlife camera-trap data is extra challenging because of the uncontrolled varying backgrounds (Ravoor & Sudarshan, 2020). Deep learning algorithms risk to learn patterns including the background, rather than a 'visual concept' of animal action that can be applied to new data (Beery et al., 2018). Especially models that perform classification based on full images often prove to be hardly applicable in practice as they are well able to do predictions for images it has seen during training but less at predicting new test images with different backgrounds or light conditions (Tuia et al., 2022).

Therefore, a type of deep learning models that lends itself to take a good look are object detection models (Michelucci, 2019). What these models do is instead of classification based on the full images, is to focussing only on the area of the animal focussing on the relevant properties (Kubat & Kubat, 2017; Michelucci, 2019). As object detection methods experience less background disturbance, these methods could provide a powerful solution against overfitting on the background, creating a model also suitable for predictions to new data (Beery et al., 2018). Goal of this study is to experiment with two leading object detection methods for Red deer (*Cervus elaphus*) action recognition and evaluate how well they generalize to new data also considering the human effort of creating the methods.

YOLOv5 (Bochkovskiy, Wang, & Liao, 2020) was used as the first method, because this model is currently very popular for object detection tasks and the model can very quickly and accurately predict a class based on a bounding box which is a rectangle fitted around the animal (Bochkovskiy et al., 2020; Michelucci, 2019). In the pose estimation approach (PEA) a random forest algorithm used key-point features that were obtained by pose estimation technique DeepLabCut (DLC), to classify behaviour (Mathis et al., 2018). Although more time-consuming to create, due to the labelling of the key-points, it was hypothesized that PEA would lead to a better predictive performance because of its ability to generate synthetic key-points and fine-grained control by only looking at the relevant animals key-points (Pereira, Tabris, et al., 2020). PEA was especially interesting because it was stated that DLC could already make accurate predictions with a limited number of training data (Nath et al., 2019; Pereira, Shaevitz, & Murthy, 2020; Pereira, Tabris, et al., 2020) while for YOLOv5 perhaps more images are needed as every permutation of each pose need to be collected.

As a case study, 506 single Red deer (*Cervus elaphus*) images, were used, collected by camera-traps scattered across National Park Hoge Veluwe, a 50-km^2 game reserve in the Netherlands. Even though action recognition based on a single image does not reflect reality in which actions are temporal and dynamic, due to time constrains and the challenge of using deep learning, this study focusses on detecting action from single Red deer (*Cervus elaphus*) images. Further research could build on this research by making action recognition also possible for multiple animals and considering consecutive images.

1.2 This study

RQ 1: How can overfitting be reduced?

RQ 2: How feasible are the two methods in terms of human effort to build and implement?

RQ 3: How do the methods perform?

1.3 Outline

This report consists of 5 chapters. The first chapter, introduction, explains the latest developments in wildlife action recognition and presents two potential automated methods: YOLOv5 and the pose estimation approach (PEA). The second chapter, methods, describes the research area, National Park Hoge Veluwe (NPHV), choices for the actions, creation of the dataset from Agouti and the set-up of the two methods. Chapter 3, results, shows for each method the predictive performance, human effort to create the models and computational training effort. As PEA contained a traditional machine learning classification algorithm, for this method also a feature importance graph was shown. Chapter 4, discussion, compares both methods and puts them into perspective to other scientific papers. In addition, limitations of both methods are reviewed and based on these, recommendations are given for future action recognition research. In the final chapter, reference list, an overview is given of the consulted literature

2. Methods

2.1 Study system

National Park Hoge Veluwe (NPHV) is a fenced nature reserve with a total area of 50 km² located in the eastern province Gelderland of the Netherlands (Appendix 1: Map National Park Hoge Veluwe). NPHV has a temperate climate with relatively mild winters and summers (Grieser, Gommes, Cofield, & Bernardi, 2006). NPHV covers 5% of the Veluwe which is the biggest contiguous nature reserve of the Netherlands and is visited by over half a million people each year. Six diverse landscapes are present, resulting from historical land use: drift sand, meadow, dry and wet heathland and two types of forest (National Park Hoge Veluwe, 2021).

NPHV houses amongst other large mammals, like mouflon (*Ovis gmelina*), roe deer (*Capreolus capreolus*), wild boar (*Sus scrofa*) and fallow deer (*Dama dama*), approximately 200 Red deer (*Cervus elaphus*) individuals. Red deer (*Cervus elaphus*) species belongs to the clade of ungulates characterized by walking on hooves. As ungulates basically walk on their toes, the heel is not connected to the ground and the knee is usually positioned at the base of the body (Janis, 1998). Red deer (*Cervus elaphus*) is an herbivorous species and their diet consist of grass, heather, leaves, buds and shoots (National Park Hoge Veluwe, 2021). It is one of the largest deer species in the world and can be found in most of Europe. The animals length is between 160cm and 250cm and their weight varies from 120kg to 240kg with males usually larger and bigger than females (National Park Hoge Veluwe, 2021). Red deer (*Cervus elaphus*) have several prominent and/or contrasting body features like slender legs, a thin face, pointy ears and a black muzzle. The orangish to brown coloured skin makes them less noticeable in their environment (Rattray, 2009).

There are throughout NPHV up to 70 permanent camera-traps that continuously record animal activity varying in viewpoint and shape. As a result of the camera-traps distributed over NPHV with at least eight camera stations per habitat type, the database includes a wide variation in habitats (National Park Hoge Veluwe, 2021). The animals are recorded with wildlife camera-traps (HC500, RECONYX, Holmen USA) that are placed on poles 70 cm above the grounds surface. The camera-trap data includes variation in lightning conditions by recording 3.1 MP colour images (RGB) during daylight and monochrome black and white images in night-time. While most cameras record images with dimensions 2048 x 1427, due to a mistake in camera settings some images have dimensions 1920 x 971. Camera-traps that detect an animal with their passive infrared (PIR) heat sensor shoots a sequence of 10 images with an image rate of 0.9 seconds. When the camera is triggered again shortly after, another 10 images are recorded and added to the previous sequence.

The NPHV Agouti dataset (agouti.eu) with content from 2013 contains half a million sequences and more than 6 million images including metadata like the URL, location, time and sequence and an unique multimedia identifier (Cameratrap DP Development Team, 2021). Volunteers, students and researchers work on labelling images, for example with the type of species, the amount of animals, sex and age (Casaer, Milotic, Liefting, Desmet, & Jansen, 2019). Since 2021 the functionality of action labelling has been added. Although it is welcomed that such a function is put into use, currently only 6 actions can be chosen which apply to all animal species, which makes it evidently insufficiently developed to use in this research.

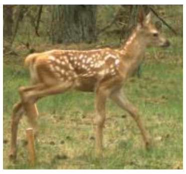
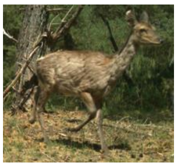
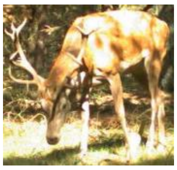
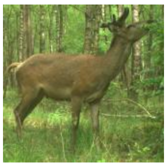
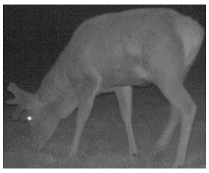
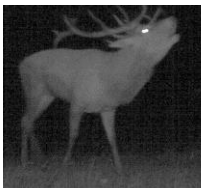
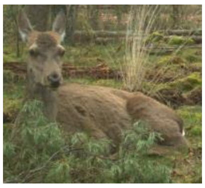
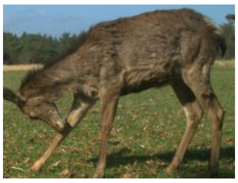
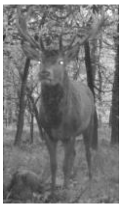
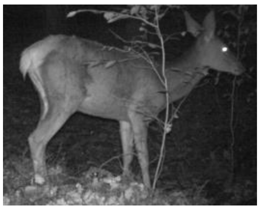
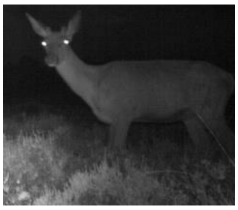
2.2 Taxonomy of actions

11 Red deer (*Cervus elaphus)* actions have been detected after image analysis and literature review. Red deer (*Cervus elaphus*) are intermediate foragers that *graze*[1] (on grasses), *browse*[2] (on hardwood vegetation) and *scan*[3] (attempting to forage) (Gebert & Verheyden-Tixier, 2001). The Red deer (*Cervus*

elaphus) gait can be distinguished by *walking*[4] generally characterized by having mostly 3 contact points with the ground and *running*[5] with maximum 2 contact points (Wada, 2022). The sixth and seventh action are *sitting*[6] on the ground and *standing*[7] upwards, in which the body is not particularly tense, and the ears are low. More often however, sitting or standing is accompanied by *vigilance*[8], in which the animal is on her guard with a tense leg position and upward pointing ears (Rattray, 2009). The nineth action is *grooming*[9] in which the animal cleans its fur to remove insects and parasites (Graystock & Hughes, 2011). Typical mating behaviour of males is *roaring*[10] (National Park Hoge Veluwe, 2021). The sound of camera-traps recording images cause a non-natural eleventh action which is when the animal is *watching into the camera*[11].

The above-described actions resulted in 11 Red deer (*Cervus elaphus*) actions (*browsing, camera watching, grazing, grooming, roaring, running, scanning, sitting, standing, vigilance and walking*) (Table 1). To ensure concise action labelling for all images during the dataset creation phase a protocol was drawn up that included all 11 actions (Gils, 2021) of which the extended labelling criteria for each behaviour are included in this report (Appendix 2: Action annotation protocol).

Table 1 The 11 different forms of Red deer (Cervus elaphus) action from https://www.agouti.eu/.

Walking 	Running 	Scanning 	Browsing 
Grazing 	Roaring 	Sitting 	Grooming 
Standing 	Vigilance 	Camera watching 	

Labelling these actions unavoidable contained several limitations. First, although it was possible to expand action running by trot, pace and gallop, it was decided not to do this because actions were chosen so that they would be immediately understood by all annotators. Second, as defecating or urinating were not observed in the dataset, these behaviours were excluded from the action protocol. In addition, social actions were excluded as this research focussed on recognizing individual actions. Third, limitation to this human labelling protocol was that inconsistencies between analyses could occur as annotators despite the extensive description of actions do not always interpret these criteria in the same way (Schneider et al., 2019). Therefore, special attention has gone out to very precisely specifying the criteria of classes which are mostly confused like are walking and running, scanning and moving, scanning and grazing/browsing and vigilance and standing. Final limitation to the action classification was that labelling images based on a single image instead of consecutive images made it more manageable to create the models, yet it reduced reliability of labelling considering actions being a dynamic and temporal activity (Pereira et al., 2019). For example, the differences in walking and running characterized by contact points was best observed from consecutive frames but had to be distinguished from a single image.

2.3 Dataset creation

The data was retrieved from Agouti (agouti.eu), a platform for managing and processing camera-trap data and filtered in R (R Core Team, 2020). As all images required action-labelling, key-point labelling and processing through the 2 approaches, due to time constraints the number of images that could be dealt with were was set to approximately 500 images. ~~Scripts are available via GitHub or the R Markdown action annotation protocol (Gils, 2021)~~. After registration and having access to the project 'the Hoge Veluwe wildlife monitoring project', unzipping the data '19 October 2021' resulted in three files: *observations.csv*, *multimedia.csv* and *deployments.csv* that were loaded into R. Filters were applied for species Red deer (*Cervus elaphus*) (1) and a single animal (2), reducing the dataset from 449575 to 6895 sequences (Table 2, part 1).

Table 2-part 1 Initial filtering for single red deer (Cervus elaphus).

	Filter	**Sequences**
	Initial Aguti dataset 2013- 2021	*449575*
1	Species Red deer	12562
2	Single Red deer	6895

To measure how the model worked for new data, independency between the train and test dataset was essential (Beery et al., 2018). Therefore, each set was assigned to a different set of camera-trap locations (3) and either to the period after 2018 or before 2019 (4) (Table 2, part 2). In addition, to prevent repetitive data within each set, the maximum number of sequences per year, per camera-trap location was set to 10 (5). Afterwards the available sequences were split into a 700 train (80%) and 175 test (20%) images, based on consulted literature (Bunkley, 2009).

Table 2-part 2 Creating independent train and test dataset

	Filter	**Train**	**test**
3	Camera-trap location	Set A	Set B
4	Year	< 2019	>= 2019
5	Maximum	Per year and per camera location max 10 sequences	Per year and per camera location max 10 sequences
		700 sequences	175 sequences

Downloading these sequences from Agouti took approximately 7 hours and files were stored per sequence identifier in a local directory. Images were deleted if these were blurry or foggy, if they despite the filter for one single Red deer (*Cervus elaphus*), still contained multiple animals and if the animal was partly outside the borders of the image (6) (Table 2, part 3). Maximum one image per sequence was selected (7) aimed to increase the number of rare actions (browsing, roaring, sitting and camera watching). Images in which Red deer (*Cervus elaphus*) was occluded by vegetation or other objects were included, allowing for the DL-models to learn predicting occluded body parts. At this stage, pre-testing the YOLOV5 bounding box detector already revealed some images despite filtering for single animal still contained multiple animals that were hard to detect with the human eye. Therefore, an extra 19 images were removed (8) ending up with a start dataset of 506 images, which the numbers matched the pre-intended goal of processing up to 500 photos. Of this start dataset 394 images belonged to the training and 112 to the test set.

Table 2-part 3 From image sequences to usable individual images

	Selection	Sequences	images
	Usable sequences	*875*	
6	Manual criteria selection	556	
7	Manual sequence to image selection		556
8	Delete multiple animals bounding box		**506 -> 394 train, 112 test**

What was not filtered out and thus preserved was variation in animals age, viewpoint and shape, habitat type and lightning conditions like day or night (Beery et al., 2018). Despite the effort to include as much variation as possible, to question whether the training dataset of 394 images with labelled 11 classes (average 35 images per class), were sufficient to make the network lean for instance animal viewpoints from different angles. Since it was stated that for the PEA technique DeepLabCut, 100 labelled images could already achieve a lower than 5 pixel error (Nath et al., 2019), the training set with 394 images seemed sufficient to obtain a good key-point prediction. However, for the action recognition, were only 35 images per class were available a problem arose because at least 150, but preferably 500 or more images per class were recommend for a reasonable accuracy (Shahinfar et al., 2020). This meant that to train a good model, only a maximum of three action classes could be made.

The issue of most classes being too small was addressed by merging either a small class with a bigger class or multiple small classes into a single large class (Figure 1). As scanning and browsing contained only respectively 43 and 7 images, these classes that above all, both related to foraging could be merged with 134 images of grazing to form the large enough and logical parent class foraging (184). Likewise, the small class running was merged with class walking forming the large enough and logical parent class moving (171). With the remaining six classes, each class with limited images, merging these created the parent class other (151) that met the minimum class size requirement of 150 images (Shahinfar et al., 2020). That all classes contained an equal amount of images was important to prevent models becoming biased towards the classes with more images (Norouzzadeh et al., 2018). The figures have been created using the R package tidyverse (Wickham et al., 2019).

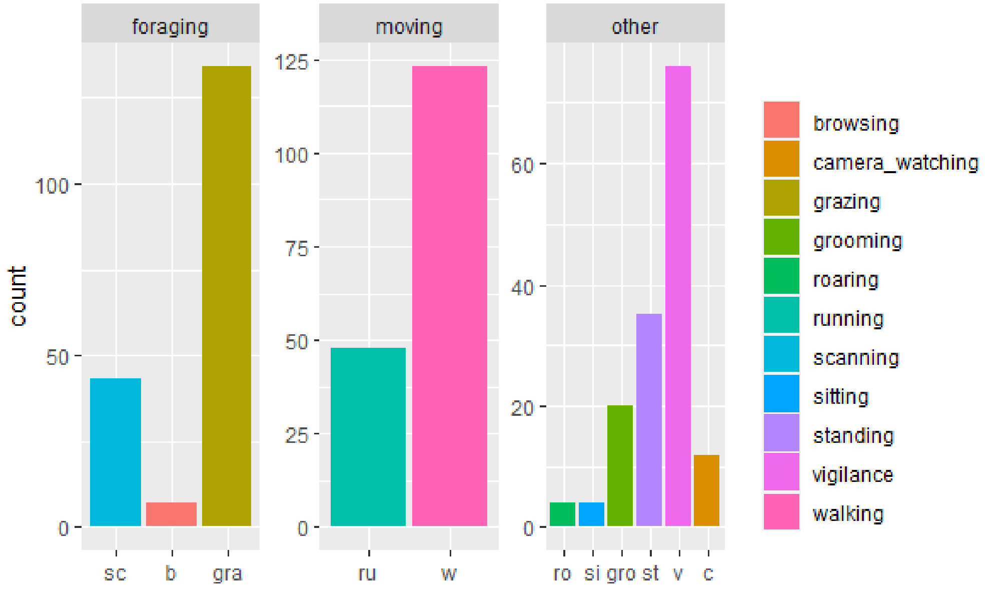
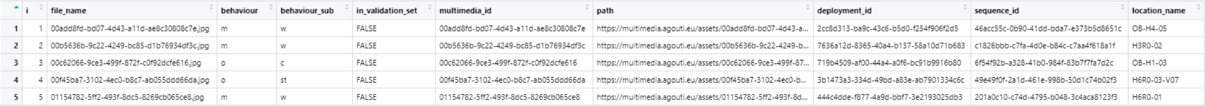


Figure 1 Distribution of the chosen Red deer (Cervus elaphus) actions (Wickham et al., 2019)

In R, all 506 images were labelled with all 11 actions and the 3 parent actions: foraging, moving and other, according to the criteria in the action annotation protocol (Gils, 2021). The result of the dataset creation was a csv file containing the filenames, action labels and the metadata (Table 3). The time effort of the dataset creation is shown in Table 4. From here programming shifted from R to Python used via the community edition of PyCharm (Van Rossum & Drake Jr, 1995), as Python had better possibilities to apply machine and deep learning.

Table 3 Result dataset creation csv file with the attributes behaviour (parent actions) and behaviour_sub (all actions).



	i	file_name	behaviour	behaviour_sub	in_validation_set	multimedia_id	path	deployment_id	sequence_id	location_name
1	1	00add8fd-bd07-4d43-a11d-ae8c30808c7e.jpg	m	w	FALSE	00add8fd-bd07-4d43-a11d-ae8c30808c7e	https://multimedia.agouti.eu/assets/00add8fd-bd07-4d43-a...	2cc8d313-ba9c-43c6-b5d0-f254f906f2d5	46acc55c-0b90-41dd-bda7-e373b5d8651c	OB-H4-05
2	2	00b5636b-9c22-4249-bc85-d1b76934df3c.jpg	m	w	FALSE	00b5636b-9c22-4249-bc85-d1b76934df3c	https://multimedia.agouti.eu/assets/00b5636b-9c22-4249-b...	7636a12d-8365-40a4-b137-58a10d71b683	c1826bbb-c7fa-4d0e-b84c-c7aa4f618a1f	H3R0-02
3	3	00c62066-9ce3-499f-872f-c0f92dcfe616.jpg	o	c	FALSE	00c62066-9ce3-499f-872f-c0f92dcfe616	https://multimedia.agouti.eu/assets/00c62066-9ce3-499f-87...	719b4509-af00-44a4-a0f6-bc91b9916b80	6f54f92b-a328-41b0-984f-83b7f7fa7d2c	OB-H1-03
4	4	00f45ba7-3102-4ec0-b8c7-ab055ddd66da.jpg	o	st	FALSE	00f45ba7-3102-4ec0-b8c7-ab055ddd66da	https://multimedia.agouti.eu/assets/00f45ba7-3102-4ec0-b...	3b1473a3-334d-49bd-a83e-ab7901334c6c	49e49f0f-2a1d-461e-998b-50d1c74b02f3	H6R0-03-V07
5	5	01154782-5ff2-493f-8dc5-8269cb065ce8.jpg	m	w	FALSE	01154782-5ff2-493f-8dc5-8269cb065ce8	https://multimedia.agouti.eu/assets/01154782-5ff2-493f-8d...	444c4dde-f877-4a9d-bbf7-3e2193025db3	201a0c10-c74d-4795-b048-3c4aca8123f3	H6R0-01

Table 4 Time effort dataset creation

Dataset creation	**Estimated time in hours**
Filter the Agouti dataset	1
Downloading Agouti images	7
Manual image selection	6
Annotating 506 images action	8
Total	*22*

2.4 Object detection

I applied and compared two object detection methods to the start dataset: the YOLOv5 and the pose estimation approach (PEA)(Figure 2).

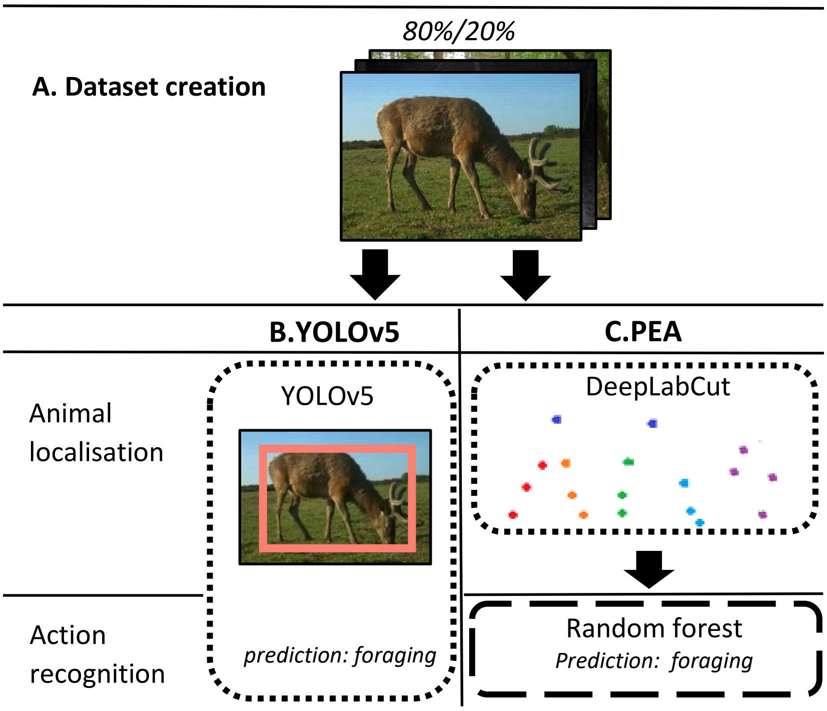


Figure 2 Overview of YOLOv5 and PEA (the pose estimation approach). Square outlines indicate a deep learning algorithm and the dashed outline a traditional machine learning algorithm

2.4.1 YOLOv5

YOLOv5 performs animal localisation and action recognition simultaneously (Michelucci, 2019). In this research YOLOv5 was applied with the GUI-driven workflow BOX21, in which users process steps by a point-and-click without the urge for programming. On several occasions, it was necessary to switch to programming with code in Python (Appendix 3: Scripts and models), because this offered more possibilities to adjust.

BOX21 was accessed by setting up a tunnelling connection to the Wildlife Ecology and Conservation GPU PC at Wageningen University's Lumen building (RTX3090 24 GB, Nvidia, Santa Clara USA) via the Any Desk software which allowed the local PC to connect to the GPU PC , on which BOX21 was running. This GPU was needed because training a deep learning neural network is a resource-intensive task, and a GPU can perform multiple, simultaneous computations. After the connection was established, BOX21 was visited by typing *localhost:8008* in the web browser. As BOX21 required a different representation of the input data, the format of the start dataset csv had to be adjusted (Appendix 4: BOX21 example input). All uploaded images were manually checked to make sure every image contained one bounding box. In addition, these bounding boxes on BOX21 had to be linked to the already annotated action labels, which was done by specifying the labels again in the configuration file, selecting bounding boxes based on 'class like' and change the bounding box labels to the original labels.

Were parameters are values internal to the model, hyperparameters are parameters that control the learning process and specifying them can improve the models learning (A. Zhang et al., 2020). The *network* was YOLOv5 and the hyperparameter *number of epochs* was tuned which described when the entire dataset was passed through the network (Hastie et al., 2009). Setting the number of epochs too low risked not having enough time to learn, but on the other hand setting it too high will risked being too adapted to the training dataset increasing on the test error. Graphs that printed the errors for the train and test dataset showed that approximately around 300 epochs was an optimum value for the number of epochs (Appendix 5: Train and test error graphs). The amount of images processed per epoch is called the *batch size* (Hastie et al., 2009). In deep learning typically minibatches are used which are equally sized subsets of the dataset. If the minibatch is set too low the network risks not reaching the optimum learning and if the minibatch is too large, the model can risk overfitting, in which the models fits the training data good, but does not generalize well on new data (A. Zhang et al., 2020). Therefore, batch size was set to the intermediate value of 16.

As the start dataset contained limited class sizes, specifying hyperparameter *image augmentation*, in which new training images were created artificially, could play an important role to prevent overfitting (Michelucci, 2019). Image augmentation methods can prevent overfitting by increasing complexity of the model during training, which produces a higher training error, but can lower the error on the test dataset (Michelucci, 2019). For YOLOv5 image augmentation argument mixup was applied, that extends the training dataset by adding combinations of images (Figure 3) (H. Zhang, Cisse, Dauphin, & Lopez-Paz, 2017). Last, the *confidence threshold* was set. Initially, the model does not outputs a class but for each class a confidence value between 0 and 1 (Norouzzadeh et al., 2018). For example, setting the confidence threshold to 1 means the model only includes predictions when it is 100% certain removing all predictions with lower confidence values. A high confidence threshold can result to no bounding box is predicted, and for that image a false negative is returned in which the model could not detect any bounding boxes. On the other hand, for a low confidence threshold, it could be that the model despite only one bounding box labelled, predicts multiple bounding boxes, which are called false positives. The confidence threshold of 0.5 was chosen to find a middle ground between not having too many false negatives and false positives. All hyperparameters were: *network: YOLOv5, epochs: 300, batch size: 16, image augmentation: mixup 0.5, confidence threshold: 0.5*

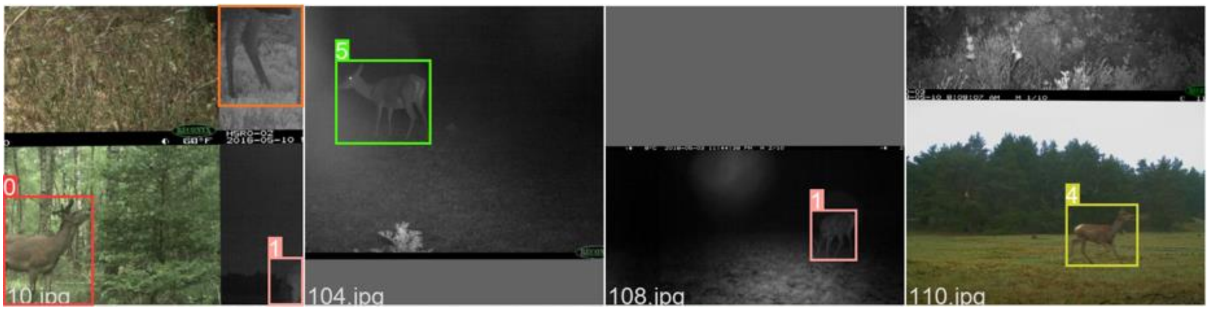


Figure 3 visual interpretation of the mixup argument in the YOLOv5 method

The first model was trained with the 3 parent actions foraging, moving and other, whereafter a second model was trained that included all 11 actions, even though it was known that this second model would be less reliable. Training took approximately 1 hour for each model and used 11.1GB GPU memory. Results were shown under the tabs most confused and models. Unfortunately obtaining individual predictions instead of the summary required some extra programming. Comparing the models' predictions with the annotated true action labels was performed in Python by printing a classification table that showed accuracy, precision and recall. Accuracy showed the percentage of correct predictions by the model, precision the percentage of a predicted class that is indeed that class and recall is the percentage of all labels of a class that have been correctly classified as that class. Precision values were also used to create a confusion matrix. All results were created with Python packages pandas and matplotlib.

The capability of applying a model to new data is called inference and could be done either via BOX21 or programming in Python. For inference via BOX21, the user can either upload new data similar as described earlier during training (Appendix 4: BOX21 example input) but leave out (or turn to False) the parameter *in validation set*. The active model can be run by navigating to *assets*, selecting the images and clicking *run model on selection*. Alternatively, download the best.pt model file from *models* for both models (Appendix 3: Scripts and models). Specify the URLs of the images you would like to predict and the path of your model file and obtain the prediction (Figure 4). The time effort of YOLOv5 is shown in

Table 12.

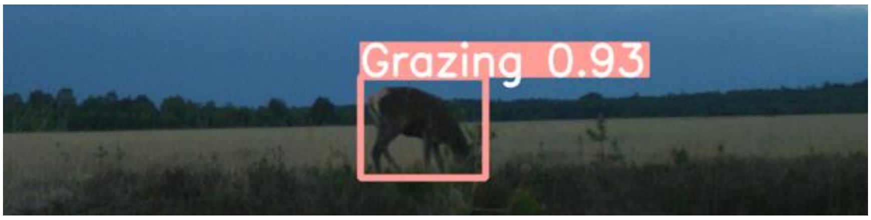


Figure 4 New data image from dataset BSc. student B. Sluiter can be classified via inference.

2.4.2.1 Box 1. Origin and how YOLOV5 works

A new family of object detection models, YOLO which stood for You Only Look Once, emerged in 2015 (Redmon, Divvala, Girshick, & Farhadi, 2016). Since 2020 the fifth version of the YOLO models, YOLOv5 is available (Bochkovskiy et al., 2020) and this network turned out to be one of the best object detection models currently in circulation. Because YOLOv5 uses the full image as context for predicting bounding boxes (Redmon et al., 2016), YOLOv5 is much more efficient compared to previous methods like Fast R-CNN that used a sliding window approach in which all proposed object regions were required to go through the neural network (Girshick, 2015). YOLOv5 is also faster and more accurate than competing object detection models like EfficientDet (Michelucci, 2019; Tan, Pang, & Le, 2020).

YOLOv5 is a convolutional neural network consisting of convolutional layers with at the end a fully connected layer. The YOLOv5 network was pretrained on the coco dataset consisting of 328.000 images (Lin et al., 2014). Applying this pre-trained network to a classification task with relatively few samples is called transfer-learning and can be useful as optimized neural network parameter values from the neural network could be also explanatory for Red deer (*Cervus elaphus*) actions (Lin et al., 2014; Pereira et al., 2019).

YOLOv5 model first divides the image into a grid and predicts for every cell in the image a class confidence of the object, like 0.87 for a foraging Red deer *(Cervus elaphus).* Then the network predicts several bounding boxes (Figure 5). The model is trained to predict the correct bounding box by evaluating the area of overlap between a predicted bounding box and a true labelled bounding box. The final detection combines the best fitting bounding box and adds the prediction of the class with the highest confidence value (Michelucci, 2019).

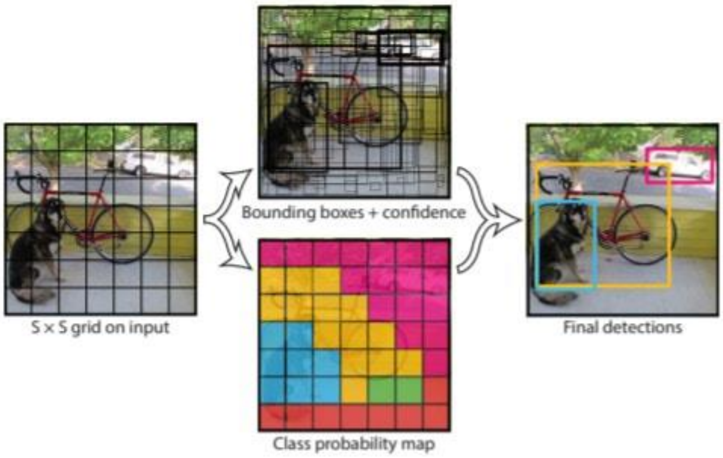


Figure 5 Schematic overview of YOLOv5

2.4.2 Pose estimation approach

The pose estimation approach (PEA) first performs animal key-point localisation using pose estimation technique DeepLabCut (DLC) followed by action recognition using a random forest algorithm.

2.4.2.1 DeepLabCut

Many online platforms for animal pose estimation software have been developed, like DeepLabCut (Mathis et al., 2018), DeepPoseKit (Graving et al., 2019), LEAP (Pereira et al., 2019) and SLEAP (Pereira, Tabris, et al., 2020). For this research DeepLabCut was chosen because of its high performance and large community to be able to ask any questions (Nath et al., 2019).

DLC was applied with a GUI-driven workflow, however, on several occasions it was necessary to switch to programming with code in Python. DLC was installed by consulting the DLC user guide (The DeepLabCut Team, 2021) and in addition the DLC protocol provided a clear overview of the complete workflow (Nath et al., 2019). After creating a project folder, the starting dataset images were uploaded via the DLC GUI. 18 key-points assumed to maximize variation of each action were chosen on parts of the body that represented joints while also being clearly visible from the outside. Visible key-points and key-points occluded by objects such as vegetation were labelled. Only key-points occluded by the animal itself, for example the knees on the non-visible side of the animal, were not labelled because the reasoning was that these key-points were often harder to guess (Figure 6) (Appendix 6: DLC annotation protocol). Labelling took approximately 2 minutes per image and thus, the total estimated time for labelling all images was 17 hours, which was performed in parts of half an hour, to ensure accurate annotation.

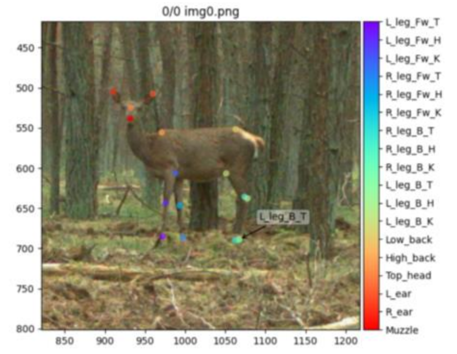


Figure 6 Annotation example from the labelling toolbox of DLC

From here, all remaining steps were run on the GPU PC. As the connection with the GPU PC via Any Desk was already established, the file-transfer software FileZilla allowed to transfer the project folder and the scripts to the GPU PC. Using the GPU PC, paths needed to be changed from *C:/Users/* to */opt/jorrit_model_rd/*, and scripts were run by using the terminal and connecting again to the GPU PC with terminal commands. Alas, the DLC-GUI did not allow for a pre-defined train and test data split. Changing the DLC code by specifying the arguments *train indices* and *test indices* of the function *create training dataset* allowed to perform the split.

Also, for pose estimation technique DeepLabCut the hyperparameters were specified. The default *network* of DLC resnet50 was chosen which in pose estimation typically consisted of an encoder and a decoder part. The encoder, extracted features from the image and the decoder used this information by predicting the key-point coordinates (Mathis et al., 2020). The *number of epochs* was set to 250.000 again based on the train- test error graphs (Appendix 5: Train and test error graphs) and the *batch size* was set to 4. Similarly to mixup used in YOLOv5, here *image augmentation* method imgaug was applied which added training data by cropping, shifting, rotating and adding contrast (Nath et al., 2019). All hyperparameters were: *network: resnet_50, epochs: 250.000, batch size: 16, image augmentation: imgaug*

Training took approximately 72 hours and used 23.4GB GPU memory. The models predicted key-points for both train and test dataset images. The margin of error was expressed in mean average error (MAE) which described the average pixel distance between the predicted key-point and its true label. With a training dataset MAE of 23.31px and a test dataset MAE of 217.45px, the model seemed to overfit quite a lot, despite using mixup. Because DL always predicts all-key-points, the white arrows in the key-point prediction plots (Figure 7) shows that DLC had trouble predicting not labelled key-points (occluded by the animal itself) as the plots show these key-points became outliers predicted on completely different body parts. Not labelling these key-points, in retrospect, probably was not a sensible annotation choice for this study and might had a considerable influence on the train and test error of the action recognition.

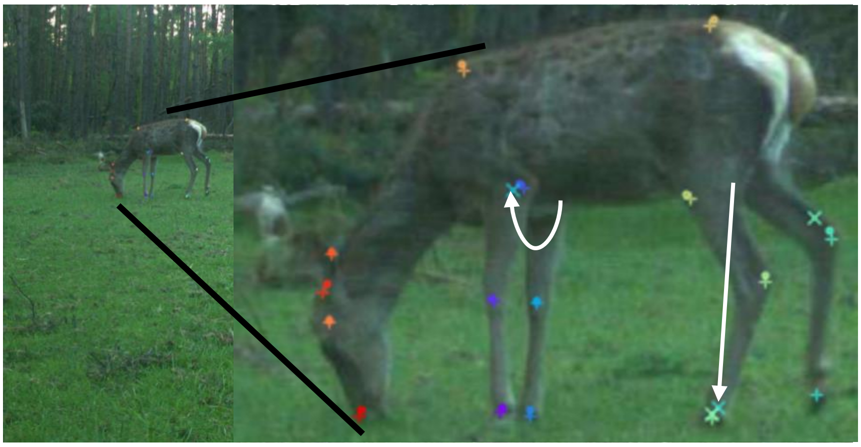


Figure 7 Example key-point prediction plot training test dataset. Labels ("+"), confident prediction ("●") and prediction("x"). White arrows shows were the key point should be and were DLC predicted the key point.

As the function analyse time lapse images, that provided the x and y coordinates of each key-point required all images to be the same size, padding was applied in which for smaller images pixels were added to have all images the same dimensions.

2.4.2.2 Random forest

From the DLC key-point coordinates key-point features were calculated in such a way that they would explain the actions foraging and moving (Figure 8). Features 1 and 2 related to the legs angle and were created by using the math package (Van Rossum, 2020) while feature 3 and 4 related to the relative distances between the head and the leg (Figure 8). A description of the key-point features and how these key-point features are distributed is shown (Appendix 7: PEA key-point features and distribution).

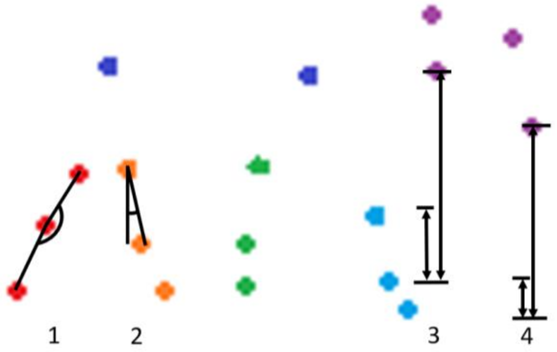


Figure 8 Visualisation of the calculated key-point features. Dots represent Red deer (Cervus elaphus) key-points.

As key-point features 3 and 4 contained large outliers, per key-point feature, values lower then 3 the third percentile were changed to the key-point features 3 percentile value and values higher then the 97th percentile were changed to the key-point features 97th percentile creating a cut-off point off 3-97% for key-point feature 3 and 4. As many algorithms perform better when the range of the values between the features are equal, the features that contained different ranges were normalised subtracting the values for each feature from its mean and scale the values to a standard deviation of 1 (Hastie et al., 2009).

Plotting a legs angle key-point feature (feature 1) and a head to leg distance key-point feature (feature 3) in relation to the left back leg for the classes moving and foraging already revealed the head to leg distance features contained more variation between the two classes (Figure 9) and thus might be more important in explaining actions.

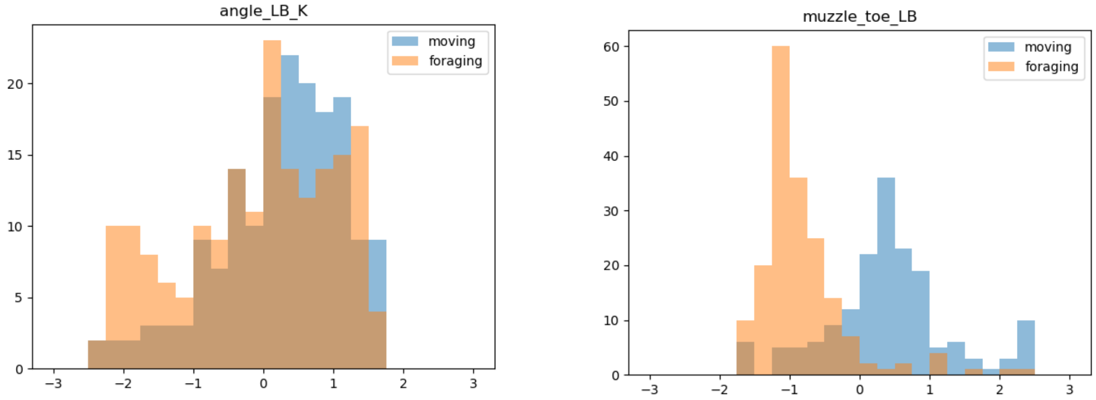


Figure 9 Histograms showing variation in distribution of the classes moving (blue) and foraging (orange) for the features relating to leg angles (left) and relative distances between head and leg (right)

There are several ways to classify an action based on key-point features. Although it is argued that deep learning has the best predictive power, it may be that for small datasets a traditional machine learning method predicts better (Freytag et al., 2016). Pre-comparison of the deep learning algorithm, multi layer perceptron (MLP) with hyperparameters *batch size=32, epochs=1000, hidden layers: 128,256,128* with a traditional machine learning random forest classifier actually confirmed literature that the traditional machine learning random forest algorithm on a small dataset can perform better than deep learning algorithms (Hastie et al., 2009). Although expected that using an MLP will outperform the random forest algorithm when up-scaling the dataset (Freytag et al., 2016), since a very small dataset was used, random forest seemed more appropriate in this case. Additional benefit using a traditional machine learning algorithm was that instead of a black box, a feature importance plot and dendrogram could be obtained which gave insight in how predictions came about.

The random forest algorithm, is an ensemble of decision trees (Hastie et al., 2009) and was applied using the Python package scikit (Pedregosa et al., 2011). A decision tree algorithm creates multiple binary splits based on optimising homogeneity of the following nodes (Kubat & Kubat, 2017).Random forest algorithms are usually more accurate than a single decision tree, because each tree contains a random selection of the training data and the final prediction is based on the class mostly predicted by all the trees (Breiman, 2001). Similarly, to YOLOv5, a model was trained for the 3 parent actions and another model for all actions. The key-point features that belonged to the training dataset (which originated from a 23.31px DLC key-point pixel error) were used as input to train the random forest recognizing actions and the key-point features that belonged to the test dataset (which originated from a 217.45px key-point pixel error) were used to test the model's performance of recognize actions.

Training took only a split second and predictions were easily obtained. The predictions were compared with the annotated true action labels by a classification table consisting of accuracy, precision and recall and a confusion matrix. In contrast to the YOLOv5 model that had a confidence threshold for the confidence of a bounding box prediction, the random forest classification algorithm was forced to always choose the class with the highest confidence, regardless of its confidence value. As YOLOv5 sometimes did not predict any category for challenging images, predictions that were likely for YOLOv5 to be predicted wrong were taken out. Therefore, the YOLOv5 model may appear better than it is. Luckily YOLOv5 only contained 2 false negatives out of 112 test images for the parent behaviour.

Inference for PEA was more challenging than YOLOv5 because the DLC model that could be applied by the function *analyze time lapse frames* contained the argument video type = '.avi' meant for videos instead of individual images. It was not clear how to change this for single images. The random forest model on the other hand, was easily downloaded and applied to new data by using the Python joblib package.

2.4.2.3 Box 2. The history of pose estimation

A new family of object detection models, called pose estimation models, emerged in 2010 outside the area of ecology (Johnson & Everingham, 2010). In pose estimation, the joints of the posture were estimated and visualized on the animals' body by small dots called key-points. For studying the action this method was a breakthrough because only a few key-points on the animals body could already reveal how the body parts were related to each other (Johansson, 1973). Pose estimation even allowed researchers to do actional predictions based on the pose like grooming or tapping for flies (Pereira, Tabris, et al., 2020). Recognition of actions with pose estimation was advantageous over other object detection approaches as it only estimated the relevant properties of objects instead of using the pixels with varying value from the object (Mathis et al., 2020). Besides fine-grained control, it was assumed that also less training data was needed to get accurate predictions (Nath et al., 2019).

Pose estimation was designed to predict human poses (Z. Cao, Simon, Wei, & Sheikh, 2017; Insafutdinov, Pishchulin, Andres, Andriluka, & Schiele, 2016; Toshev & Szegedy, 2014; Xiao, Wu, & Wei, 2018) and since 2018, has become available for animals (Pereira, Shaevitz, et al., 2020). For example, pose estimation was used for animals to monitor head action of fish (Huang, He, Wang, & Shen, 2021) and locomotion of cattle (X. Li et al., 2019) and turkeys (Straat, 2020).

Only in recent years pose estimation has been applied on wild animals in captivity, for instance to indoor living Macaques (Bala et al., 2020) and tigers living in national parks (S. Li, Li, Tang, Qian, & Lin, 2019). From a zoo in San Diego it is known that researchers are still struggeling with the question how pose estimation can contribute to automatic recognition of actions (Duhart, Dublon, Mayton, Davenport, & Paradiso, 2019). As far as is known, one of the first times that pose estimation was applied to wildlife recorded in their natural environment, the pose of a giraffe was predicted (Pereira et al., 2019). Pose estimation for wildlife is a field with enormous potential however to date, there are no known wildlife studies that have performed pose estimation followed by action recognition.

2.4.3 Background

2.4.3.2 Box 3. The history of deep learning

Early computer vision classification and object detection models mainly relied on part-based models that used parts of images separately to determine the class of an object (Fergus et al., 2003) (Figure 10). For instance, to classify the head of an animal, these models would detect the mouth, the nose and the eye, manually calculate feature distances and from here determine the class by setting manual classification rules. Besides it that it was a time consuming approach, the method consisted mainly of manual steps, in which researchers created a large bias by making assumptions, about what features were important (Schneider et al., 2019).

Machine learning deals with models that have the ability to learn from the data using algorithms without being explicitly programmed (A. Zhang et al., 2020). This means that machine learning algorithms can learn recognizing patterns and use this information to predict new unseen data (Figure 10) (Kotsiantis, Zaharakis, & Pintelas, 2007).

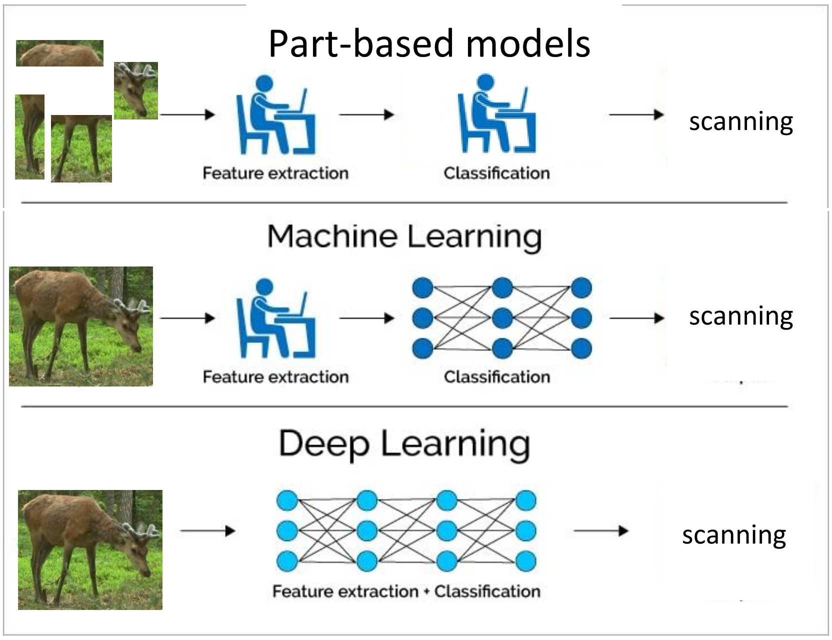
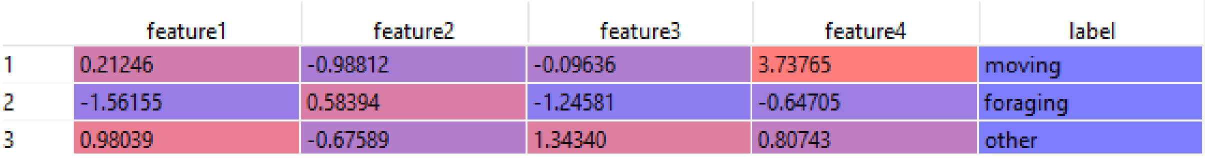


Figure 10 Schematic representation of the emergence of deep learning Source: https://www.softwaretestinghelp.com/data-mining-vs-machine-learning-vs-ai/

In supervised machine learning, like the random forest method from the pose estimation approach, many pairs of features and their corresponding labels, are used to train a model, with the features as input and the true label (e.g., “foraging)” as output (Table 5) (Kotsiantis et al., 2007; Norouzzadeh et al., 2018). Machine learning contains a wide variety of methods from regression to random forest to support vector machine (Kubat & Kubat, 2017). For recognizing actions, using machine learning was advantageous because even the position of occluded body parts could be predicted as it had learned these during training. The pre-determined features however, such as body part distances, that still required ‘hard-coding’ by programmers still risked bias and were often not applicable to another taxa (Hiby et al., 2009).

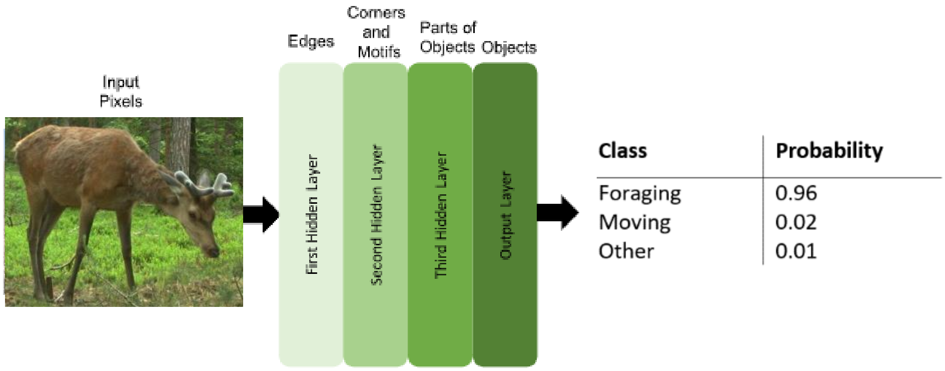
Table 5 Example machine learning dataset with features (input) and labels (output)



	feature1	feature2	feature3	feature4	label
1	0.21246	-0.98812	-0.09636	3.73765	moving
2	-1.56155	0.58394	-1.24581	-0.64705	foraging
3	0.98039	-0.67589	1.34340	0.80743	other

Deep learning evolved from machine learning and was different from traditional machine learning techniques by the multi-layered structure of their neural network that attempted to simulate the action of the human brain (A. Zhang et al., 2020) (Figure 10). In deep learning even the feature extraction was automated creating an end-to-end framework from image to classification without the need for human interference. These features that the model created automatically could be for example the presence of small- or large objects, the position of objects in the image or colour contrast. Despite the abstract concept of automatic feature extraction in which the user did not get to see what the actual features were, automatic feature extraction was a big breakthrough because it eliminated the human bias and made vast amounts of information more easily available for ecologists (Norouzzadeh et al., 2018). Due to this advantage, deep learning often improved performance over machine learning methods especially for large datasets (Freytag et al., 2016; Schneider et al., 2019).

Neural networks exist of multiple layers of which each layer of the builds upon the previous layer to optimize predictions. The most basic form is the multi layered perceptron (MLP) containing, one input, one hidden and one output layer. Even the MLP, the most simple form of a neural network can already make approximate predictions (Kratsios, 2021) as activation functions are used that add non linearity that could theoretically represent any function to fit the data (A. Zhang et al., 2020). This basic neural network does not work for images but is very suitable to recognize actions based on key-point features like in PEA. Convolutional neural networks (CNN) are a class of neural networks with multiple hidden layers in which for each hidden layer filters are applied to extract information from overlapping regions that turn out to work particularly well for images (Michelucci, 2019; Norouzzadeh et al., 2018). In a CNN, Pixels represented by numerical values are inserted into the network and in each layer, features are abstracted from big details in the first layers like the edge of the back, to more smaller details like an eye, in the last layer (Figure 11). Last, a SoftMax function forces to predict for each action a confidence value between 0 and 1. Depending on the algorithm some like YOLOv5 determines the returned prediction based on a confidence threshold, were other algorithms like random forest has the default settings to force choosing prediction even if all confidence values are low. Challenges when training a deep learning network are that they generally require a lot more input data compared to a machine learning model because neural networks consist of many parameters (Michelucci, 2019; Pereira, Shaevitz, et al., 2020).



Class	Probability
Foraging	0.96
Moving	0.02
Other	0.01

Figure 11 Schematic overview of a convolutional neural network. Learned features are not human-specified but learned by the network (Norouzzadeh et al., 2018).

2.4.3.2 Box 4. Automated action recognition of wildlife

Most deep learning algorithms have been developed outside the field of ecology in computer image recognition, security or social media industries (Schneider et al., 2019). It is common for developments in camera recognition to shift from controlled and closed set environments like the laboratory to more uncontrolled environments like wildlife (Ravoor & Sudarshan, 2020). Wildlife cameras have to deal with variation in animals viewpoint, image quality, occlusion, dynamic background, weather conditions, seasonality and an unknown number of individuals from different species that might enter and leave the camera at different times (Pereira et al., 2019).

Although criticism that deep learning for wildlife has not been widely used yet (Schneider et al., 2019), over the last couple of years turtles (Carter, Bell, Miller, & Gash, 2014), primates (Brust et al., 2017; Freytag et al., 2016; Loos & Ernst, 2013) and elephants (Körschens, Barz, & Denzler, 2018) were identified with deep learning for conservation purposes. Action recognition however can be much more challenging than identification as actions mostly deal with more small-scale differences (Nath et al., 2019).

During a first attempt to classify wildlife action, in the African savanna system researchers used the AlexNet model to predict the presence or non-presence of 6 actions. Unfortunately, the classification was based on the entire image and looking at the predictions it seemed to occur frequently that although actions were predicted by the model, these were not always clear from the image (Norouzzadeh et al., 2018). Schindler & Steinhage on the other hand used 477 Red deer (*Cervus elaphus*) media files and the object detection model ResNet, to also classify 3 action classes from Red deer (*Cervus elaphus*) with 63,8% precision (Schindler & Steinhage, 2021).

Object detection methods that classify wildlife actions based on a animal localisation can prevent overfitting, an advantage that the researchers who used the ResNet object detection model probably took into account (Kubat & Kubat, 2017; Michelucci, 2019). To know how YOLOv5 and PEA would perform compared to the method used by Schindler & Steinhage (2001), it was needed to use the same input dataset, which unfortunately, was not feasible in this study. Nevertheless, it does seem that a 63.8% precision is on the low side making it interesting to see at how accurate the YOLOv5 and pose estimation approach can predict wildlife action in general.

3. Results

For recognition of the parent actions foraging, moving and other, the overall accuracy, which is the percentage of correct predictions by the model, differed between the two approaches (Figure 12**Error! Reference source not found.Error! Reference source not found.**). YOLOv5 predicted 98 % of the test dataset (2 false negatives) with 77% accuracy and for the pose estimation approach, all images were predicted with an accuracy of 53%.

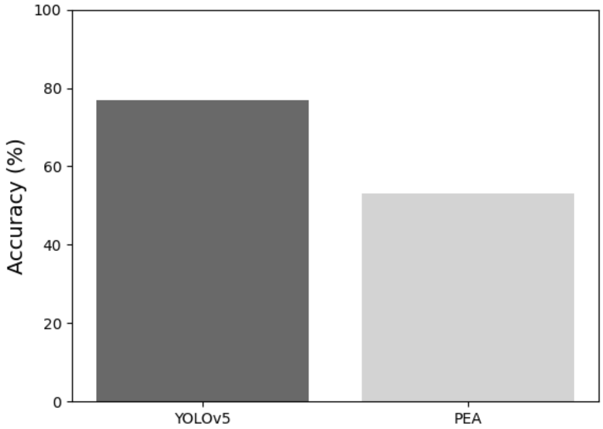


Figure 12 Overall accuracy YOLOv5 and PEA (pose estimation approach)

The precision, the percentage of a class predictions that is indeed that class (Kubat & Kubat, 2017) was substantially higher for YOLOv5 than for PEA (Figure 13). Both methods had most difficulty predicting the class other, probably due to its heterogeneity. Although there were hardly any differences between foraging and moving for YOLOv5, for PEA, moving predictions were significantly worse than foraging. Moreover the recall, the percentage of a class that had been correctly classified as that class, was much lower for moving (33%) than foraging (76%) (Figure 13) (Kubat & Kubat, 2017). It is very likely that, not labelling key-points occluded by the animal has led to erroneous knee key-point predictions which influenced the leg angle features explanatory for the walking actions. Alternatively, maybe since walking related to small scale movement of the legs and foraging to larger scale movement of the head PEA had more difficulty detecting small scale details compared to YOLOv5 in which no difference large differences in precision were observed. YOLOv5's low recall for moving (66%), however, showed that the model had difficulty recognizing moving images in the dataset.

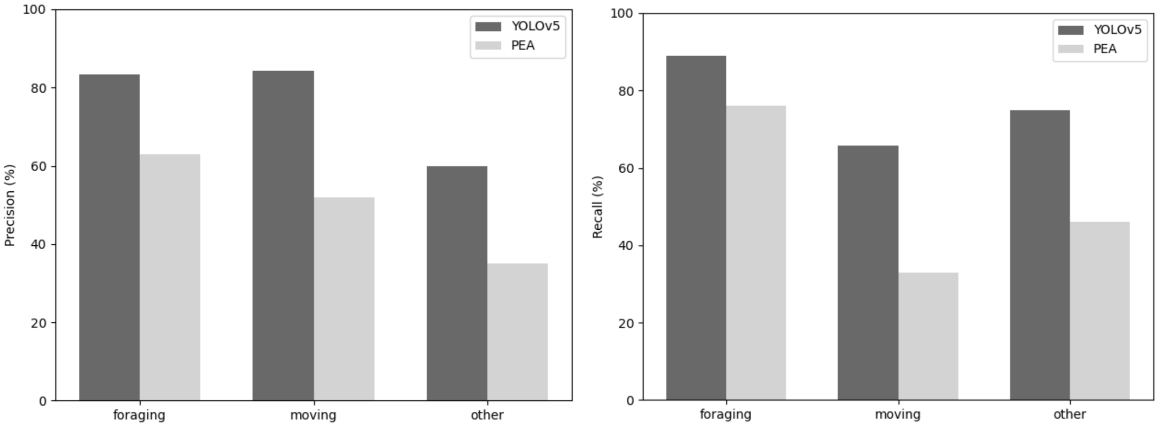


Figure 13 Precision (left) and recall (right) YOLOv5 and PEA (pose estimation approach)

The confusion matrixes summarize the classes to which the predictions were mostly confused by breaking down the predictions for each class (Figure 14). Ideally, the matrix would have a black diagonal line in which all predicted actions match the true classes. It is noticeable that a significant part of the images YOLOv5 predicted as other were truly moving, while images predicted as moving were rarely predicted as other. Again, is assumed that because other was such a heterogeneous group, predicting this class could lead to confusions easily, in contrast to moving which mostly consisted of walking images. Mismatches in PEA occurred in all classes, which shows the model was confusing the classes with each other.

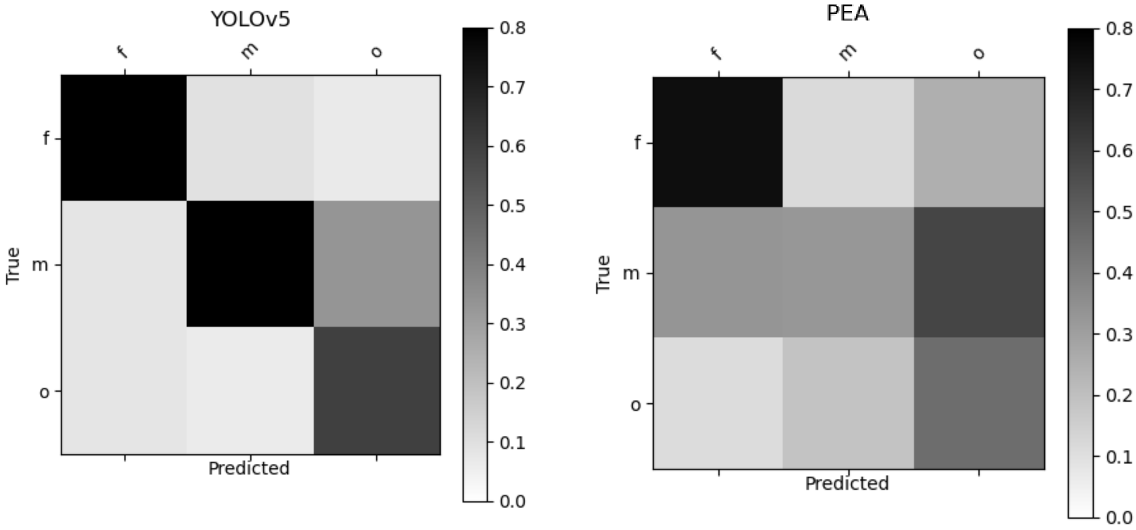


Figure 14 confusion matrix YOLOv5 (left) and PEA (pose estimation approach) (right). Predicted action (x-axis) and true labelled action (y-axis)

Since in PEA, the random forest belonged to the traditional machine learning methods, a feature importance graph was shown indicating which features were most explanatory for action recognition (Figure 15). It used Mean Decrease in Impurity (MDI) as a measure of how much a splitting criterion gained by including this variable (Breiman, 2001). Key-point features related to the angles of the legs turned out to be less informative than features dealing with distances between the head and the leg which as mentioned before was not remarkable as these features were strongly influenced by incorrect key-point predictions.

Similar findings were found in the dendrogram of a decision tree that used the predictions of the random forest as input. This dendrogram showed that head to leg features were used to distinguish foraging frames from the classes moving and other, and that a split by the leg angle feature was not as effective to separate large, homogenized groups (Appendix 8: Results).

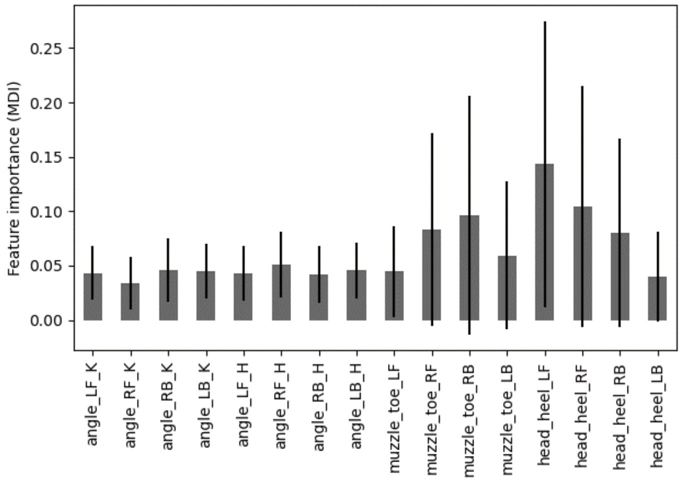


Figure 15 Feature importance random forest (PEA). 1-8 leg angle features, 9-18 distance head to leg features

For the models that looked at all 11 actions, YOLOv5 only predicted 92% of the test dataset and only with 50% accuracy (Appendix 8: ResultsAppendix 8: Results). For PEA all images were predicted but only with an accuracy of 36%. Both methods therefore had considerably more difficulty predicting all behaviours. For the classes with a relatively large number of images, grazing, scanning, walking, vigilance and standing, most of the time again YOLOv5 performed better than PEA. For the classes with a relatively few images, browsing, running, sitting, roaring, grooming and camara watching, hardly any of these categories were predicted which was not surprising as imbalanced datasets tend to bias towards classes with more examples (Norouzzadeh et al., 2018)

Also looking at the cost in terms of time it took to create, train and apply the models, YOLOv5 outperformed PEA (Table 6). PEA took 35 more hours to create the model mainly due to labelling of key-points, 69 more hours to run the model and extra time to apply the DLC model to new data. Finally, PEA needed 12,3 GB more GPU memory.

Table 6 Costs of the two modelling approaches for actional classification

	YOLOv5	**Pose estimation approach**
Time effort creating model excl. data collection	**7 (-35)**	42 (+35)
Runtime model	**1 (-69)**	DLC 70 + RF 0 (+69)
Inference hours	**1 (+)**	>1 (-)
GPU memory	**11.1 GB (-12,3)**	DLC 23.4 GB RF 0 (+12,3)

4. Discussion

Camera-traps provide snapshots of animals engaged in a variety of actions, which can inform ecology and management (Norouzzadeh et al., 2018), but manual annotation of imagery is laborious (Schneider et al., 2019). The aim of this thesis was to find out whether and how machine learning techniques can automate action recognition. I compared two methods, YOLOv5 (Bochkovskiy et al., 2020) and the pose estimation approach (PEA) consisting of pose estimation technique DeepLabCut (Mathis et al., 2018) followed by a random forest (Kubat & Kubat, 2017). I found that YOLOv5 is 15,9% more accurate than PEA and with less effort creating the methods.

YOLOv5 was excellent at predicting moving (precision 84%) and foraging (precision 83%) images yet the lower recall of moving (recall 66%) compared to foraging (recall 89%) showed its difficulty detecting moving frames in the dataset. Class other images were significantly worse predicted (precision 60%) but with a higher recall (recall 75%). As other YOLOv5 can achieve a 99,3% average precision with a dataset of at least 400 images per class, it is likely current results can be improved by adding more training data (Liu, Tang, & Zou, 2021; Shahinfar et al., 2020). More training data is especially needed when the focus switches from the 3 parent behaviours to all behaviours, were finer details come into play (Bochkovskiy et al., 2020). Comparable action recognition research in which YouTube videos were used as input to a I3D convolutional network achieved a much lower accuracy of 36%, but was also more challenging as it was designed to recognize 3 or 4 actions of 32 animal species (W. Li et al., 2020).

Looking at the precision and recall PEA performed reasonable predicting class moving, but much worse predicting classes moving and other. As in PEA precision ranges from 35% to 63%, it is clear that in this study, the supposed benefits for pose estimation such as fine grained control and good performance on small datasets with the current set-up were not found (Nath et al., 2019). The pixel error which is 217.45px and could be 5px indicates that a large amount of accuracy has already been lost in predicting key-points (Nath et al., 2019). In related work were, using a neural network, human actions were classified based on pose-key-points, a 90% accuracy was reached which show that PEA has not been used to its full potential (Song, Zhang, Shan, & Wang, 2020).

YOLOv5 and PEA were compared by accuracy, precision, recall and human effort. Comparison showed YOLOv5 achieved a better performance particularly for action foraging with less effort. In comparison to the action recognition study of Schindler and Steinhage who obtained 63.8% precision also on classifying 3 Red deer actions, YOLOv5 seems to perform quite a lot better and PEA much worse, however, it should be noted in this comparison that a different dataset and a segmentation network ResNet in which the exact boundaries of the objects are determined were used (Schindler & Steinhage, 2021).

Extra attention was paid to the prevention of overfitting by choosing object detection methods that used animal localisation instead of directly classifying from the full image. Moreover, variation in camera-trap locations, age, habitats, animals shape, animal viewpoint and light conditions was preserved to allow for a better generalization to new data (Beery et al., 2018). Similarly, image augmentation methods mixup and imgaug that artificially increased training data, have likely contributed to preventing overfitting by adding complexity to the training process and lowering the error on the test dataset (Michelucci, 2019). Particularly the good results of YOLOv5s class foraging shows the model has been able to predict new data correctly. Further research would be needed to better understand how the individual overfitting measures have contributed to a better generalization.

A limitation to YOLOv5 was that 2 false negative images did not contain any prediction, so that analysis on the number of wrong predictions turned out slightly more positive for YOLOv5 than it was. YOLOv5 could be improved by increasing the dataset and testing different hyperparameter values like number of epochs and batch size. Although these measures are also expected to improve PEA, for PEA, there were also several other presumed causes for the lower performance in PEA.

First, DLC was trained on full images and with a MAE 217.45px was likely to have resulted in overfitting by background disturbance (Michelucci, 2019). Second, in hindsight not annotating key-points occluded by the animal was a handicap, as these key-points were often predicted on other body parts affecting key-point features especially related to the leg angle important for moving actions. Third, while the random forest algorithm outperformed alternatives such as MLP, maybe an alternative deep learning action classifier with optimized hyperparameters is more effective.

A limitation to this study in general was optimal hyperparameter values were obtained by running methods several times, making the network good at predicting the test dataset but less for predicting new data. Another limitation was that due to the predetermined train and test split, the results of this study have not been validated by cross validation, a resampling method using different proportions of the data for training and testing on different iterations, which can show results are not out of “luck”, but constantly performs around a certain accuracy (Kubat & Kubat, 2017).

As deep learning for wildlife action recognition was not much explored yet (W. Li et al., 2020), this research bridged the gap and brought major innovations. Ecologists now can easily access large volumes of behaviour data for Red deer (Cervus elaphus) or any other animal with little effort. Although method PEA still needs improvement, ecologists can directly apply the robust YOLOv5 model for instance to determine feeding behaviour of wild livestock (Oliveira, Pereira, Bresolin, Ferreira, & Dorea, 2021), recognize migration patterns in response to climate change (Davidson et al., 2020), detect abnormalities in responses to poachers (Eikelboom, 2021) or estimate daily activity patterns (Rowcliffe, Kays, Kranstauber, Carbone, & Jansen, 2014) and determine how these differ from the global activity pattern (Hester, Gordon, Baillie, & Tappin, 1999; Jayakody, Sibbald, Gordon, & Lambin, 2008; Rattray, 2009).

It is still worth doing PEA again, but it makes sense to use the animal's bounding box as input for DLC, always annotate all visible and occluded key-points and add additional key-point features especially when classifying all actions. An additional advantage of taking the bounding box, which is already present in YOLOv5, is that classification per bounding box is possible and can therefore apply action recognition to an image with several animals. Moreover it is recommended to explore deep learning alternatives for action recognition such as the graphical convolutional neural network ResGCN, which also includes a key-point skeleton and can classify actions based on consecutive images (Song et al., 2020). It would be interesting to see whether YOLOv5 is still better when looking at consecutive images. Maybe YOLOv5 just provides a classification for each individual image, were ResGCN include information on key-point change from image to image (H. Cao et al., 2021).

In general, to prove that the results are statistically sound it is recommended to add cross validation to both methods. To reduce overfitting, it is recommended to add a validation set used to find optimal values for the hyperparameters which can then be applied to the test dataset (Van Etten, Lindenbaum, & Bacastow, 2018). To increase performance is highly recommended, whether the aim is to classify the parent actions or all actions, to use at least 500 images per class to obtain a good accuracy (Norouzzadeh et al., 2018; Shahinfar et al., 2020). Action labelling effort can be shortened by implementing all animal species actions into Agouti so that people can collectively contribute to a database of action annotated training data. Explore the possibilities of expanding into multiple actions of which this study already contains the protocols for action and key-point labelling, so that it is only necessary to expand the dataset and determine key-point features. Both methods can be easily trained on data from other animals, whereby only the key-points (features) of PEA must be determined again.

5. Reference list

Altmann, J. (1974). Observational study of behavior: sampling methods. *Behaviour, 49*(3-4), 227-266.

Bala, P. C., Eisenreich, B. R., Yoo, S. B. M., Hayden, B. Y., Park, H. S., & Zimmermann, J. (2020). Automated markerless pose estimation in freely moving macaques with OpenMonkeyStudio. *Nature communications, 11*(1), 1-12.

Beery, S., Van Horn, G., & Perona, P. (2018). *Recognition in terra incognita.* Paper presented at the Proceedings of the European conference on computer vision (ECCV).

Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*.

Braude, S., Margulis, S., & Broder, E. D. (2017). The Study of Animal Behavior Provides Valuable Opportunities for Original Science Fair Projects: Recommendations from The Animal Behavior Society, Education Committee. *The American Biology Teacher, 79*(6), 438-441.

Breiman, L. (2001). Random forests. *Machine learning, 45*(1), 5-32.

Brust, C.-A., Burghardt, T., Groenenberg, M., Kading, C., Kuhl, H. S., Manguette, M. L., & Denzler, J. (2017). *Towards automated visual monitoring of individual gorillas in the wild.* Paper presented at the Proceedings of the IEEE International Conference on Computer Vision Workshops.

Bunkley, N. (2009). Joseph Juran," Pioneer in Quality Control, Dies. *New York Times, 103*.

Cameratrap DP Development Team. (2021). Camera Trap Data Package. Retrieved from https://tdwg.github.io/camtrap-dp/

Cao, H., Xu, Y., Yang, J., Mao, K., Yin, J., & See, S. (2021). Effective action recognition with embedded key point shifts. *Pattern Recognition, 120*, 108172.

Cao, Z., Simon, T., Wei, S.-E., & Sheikh, Y. (2017). *Realtime multi-person 2d pose estimation using part affinity fields.* Paper presented at the Proceedings of the IEEE conference on computer vision and pattern recognition.

Carter, S. J., Bell, I. P., Miller, J. J., & Gash, P. P. (2014). Automated marine turtle photograph identification using artificial neural networks, with application to green turtles. *Journal of experimental marine biology and ecology, 452*, 105-110.

Casaer, J., Milotic, T., Liefting, Y., Desmet, P., & Jansen, P. (2019). Agouti: A platform for processing and archiving of camera trap images. *Biodiversity Information Science and Standards*.

Davidson, S. C., Bohrer, G., Gurarie, E., LaPoint, S., Mahoney, P. J., Boelman, N. T., . . . Jennewein, J. (2020). Ecological insights from three decades of animal movement tracking across a changing Arctic. *Science, 370*(6517), 712-715.

Duhart, C., Dublon, G., Mayton, B., Davenport, G., & Paradiso, J. A. (2019). *Deep learning for wildlife conservation and restoration efforts.* Paper presented at the 36th International Conference on Machine Learning, Long Beach.

Eikelboom, J. A. (2021). *Sentinel animals: Enriching artificial intelligence with wildlife ecology to guard rhinos.* Wageningen University and Research,

Fergus, R., Perona, P., & Zisserman, A. (2003). *Object class recognition by unsupervised scale-invariant learning.* Paper presented at the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.

Freytag, A., Rodner, E., Simon, M., Loos, A., Kühl, H. S., & Denzler, J. (2016). *Chimpanzee faces in the wild: Log-euclidean CNNs for predicting identities and attributes of primates.* Paper presented at the German Conference on Pattern Recognition.

Gebert, C., & Verheyden-Tixier, H. (2001). Variations of diet composition of red deer (Cervus elaphus L.) in Europe. *Mammal Review, 31*(3-4), 189-201.

Gils, J. v. (2021). Behaviour annotation protocol for camera-trap data in R. Retrieved from https://rpubs.com/JorritvanGils/BehaviourProtocolR

Girshick, R. (2015). *Fast r-cnn.* Paper presented at the Proceedings of the IEEE international conference on computer vision.

Graving, J. M., Chae, D., Naik, H., Li, L., Koger, B., Costelloe, B. R., & Couzin, I. D. (2019). DeepPoseKit, a software toolkit for fast and robust animal pose estimation using deep learning. *Elife, 8*, e47994.

Graystock, P., & Hughes, W. O. (2011). Disease resistance in a weaver ant, Polyrhachis dives, and the role of antibiotic-producing glands. *Behavioral Ecology and Sociobiology, 65*(12), 2319-2327.

Grieser, J., Gommes, R., Cofield, S., & Bernardi, M. (2006). New gridded maps of Koeppen's climate classification. *Data, methodology and gridded data Available at: http://www. fao. org/nr/climpag/globgrids/KC_classification_en. asp. Methodology also downloadable from http://www. juergen-grieser. de/downloads/Koeppen-Climatology/Koeppen_Climatology. pdf (accessed September 2015)*.

Hastie, T., Tibshirani, R., Friedman, J. H., & Friedman, J. H. (2009). *The elements of statistical learning: data mining, inference, and prediction* (Vol. 2): Springer.

Hester, A., Gordon, I., Baillie, G., & Tappin, E. (1999). Foraging behaviour of sheep and red deer within natural heather/grass mosaics. *Journal of Applied Ecology, 36*(1), 133-146.

Hiby, L., Lovell, P., Patil, N., Kumar, N. S., Gopalaswamy, A. M., & Karanth, K. U. (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.

Huang, Z.-J., He, X.-X., Wang, F.-J., & Shen, Q. (2021). A Real-Time Multi-Stage Architecture for Pose Estimation of Zebrafish Head with Convolutional Neural Networks. *Journal of Computer Science and Technology, 36*(2), 434-444.

Insafutdinov, E., Pishchulin, L., Andres, B., Andriluka, M., & Schiele, B. (2016). *Deepercut: A deeper, stronger, and faster multi-person pose estimation model.* Paper presented at the European Conference on Computer Vision.

Janis, K. M. (1998). *Evolution of tertiary mammals of North America: Volume 1, terrestrial carnivores, ungulates, and ungulate like mammals* (Vol. 1): Cambridge University Press.

Jayakody, S., Sibbald, A. M., Gordon, I. J., & Lambin, X. (2008). Red deer Cervus elephus vigilance behaviour differs with habitat and type of human disturbance. *Wildlife biology, 14*(1), 81-91.

Johansson, G. (1973). Visual perception of biological motion and a model for its analysis. *Perception & psychophysics, 14*(2), 201-211.

Johnson, S., & Everingham, M. (2010). *Clustered Pose and Nonlinear Appearance Models for Human Pose Estimation.* Paper presented at the bmvc.

Körschens, M., Barz, B., & Denzler, J. (2018). Towards automatic identification of elephants in the wild. *arXiv preprint arXiv:1812.04418*.

Kotsiantis, S. B., Zaharakis, I., & Pintelas, P. (2007). Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering, 160*(1), 3-24.

Kratsios, A. (2021). The universal approximation property. *Annals of Mathematics and Artificial Intelligence, 89*(5), 435-469.

Kubat, M., & Kubat. (2017). *An introduction to machine learning* (Vol. 2): Springer.

Lehner, P. N. (1992). Sampling methods in behavior research. *Poultry science, 71*(4), 643-649.

Li, S., Li, J., Tang, H., Qian, R., & Lin, W. (2019). ATRW: a benchmark for Amur tiger re-identification in the wild. *arXiv preprint arXiv:1906.05586*.

Li, W., Swetha, S., & Shah, M. (2020). Wildlife Action Recognition using Deep Learning.

Li, X., Cai, C., Zhang, R., Ju, L., & He, J. (2019). Deep cascaded convolutional models for cattle pose estimation. *Computers and Electronics in Agriculture, 164*, 104885.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., . . . Zitnick, C. L. (2014). *Microsoft coco: Common objects in context.* Paper presented at the European conference on computer vision.

Liu, X., Tang, G., & Zou, W. (2021). *Improvement of Detection Accuracy of Aircraft in Remote Sensing Images Based on YOLOV5 Model.* Paper presented at the 2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS.

Loos, A., & Ernst, A. (2013). An automated chimpanzee identification system using face detection and recognition. *EURASIP Journal on Image and Video Processing, 2013*(1), 1-17.

Mathis, A., Mamidanna, P., Cury, K. M., Abe, T., Murthy, V. N., Mathis, M. W., & Bethge, M. (2018). DeepLabCut: markerless pose estimation of user-defined body parts with deep learning. *Nature neuroscience, 21*(9), 1281-1289.

Mathis, A., Schneider, S., Lauer, J., & Mathis, M. W. (2020). A primer on motion capture with deep learning: principles, pitfalls, and perspectives. *Neuron, 108*(1), 44-65.

Mellor, D. J., Beausoleil, N. J., & Stafford, K. J. (2004). *Marking amphibians, reptiles and marine mammals: animal welfare, practicalities and public perceptions in New Zealand*: Department of Conservation Wellington.

Mench, J. (1998). Why it is important to understand animal behavior. *ILAR journal, 39*(1), 20-26.

Michelucci, U. (2019). *Advanced applied deep learning: convolutional neural networks and object detection*: Springer.

Nath, T., Mathis, A., Chen, A. C., Patel, A., Bethge, M., & Mathis, M. W. (2019). Using DeepLabCut for 3D markerless pose estimation across species and behaviors. *Nature protocols, 14*(7), 2152-2176.

National Park Hoge Veluwe. (2021). Red deer, the king of the park. Retrieved from https://www.hogeveluwe.nl/en/discover-the-park/nature-and-landscape/red-deer

Norouzzadeh, M. S., Nguyen, A., Kosmala, M., Swanson, A., Palmer, M. S., Packer, C., & Clune, J. (2018). Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning. *Proceedings of the National Academy of Sciences, 115*(25), E5716-E5725.

Oliveira, D. A. B., Pereira, L. G. R., Bresolin, T., Ferreira, R. E. P., & Dorea, J. R. R. (2021). A review of deep learning algorithms for computer vision systems in livestock. *Livestock Science, 253*, 104700.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Dubourg, V. (2011). Scikit-learn: Machine learning in Python. *the Journal of machine Learning research, 12*, 2825-2830.

Pereira, T. D., Aldarondo, D. E., Willmore, L., Kislin, M., Wang, S. S.-H., Murthy, M., & Shaevitz, J. W. (2019). Fast animal pose estimation using deep neural networks. *Nature methods, 16*(1), 117-125.

Pereira, T. D., Shaevitz, J. W., & Murthy, M. (2020). Quantifying behavior to understand the brain. *Nature neuroscience, 23*(12), 1537-1549.

Pereira, T. D., Tabris, N., Li, J., Ravindranath, S., Papadoyannis, E. S., Wang, Z. Y., . . . Falkner, A. L. (2020). SLEAP: multi-animal pose tracking. *bioRxiv*.

Prokhorov, A. M. l. (1970). Great Soviet Encyclopedia.

R Core Team. (2020). R: A language and environment for statistical computing. R Foundation for Statistical Computing. Vienna, Austria. Retrieved from https://www.R-project.org/

Rattray, P. (2009). *Red Deer Hunting: A Complete Guide*: Paul Rattray.

Ravoor, P. C., & Sudarshan, T. (2020). Deep Learning Methods for Multi-Species Animal Re-identification and Tracking–a Survey. *Computer Science Review, 38*, 100289.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). *You only look once: Unified, real-time object detection.* Paper presented at the Proceedings of the IEEE conference on computer vision and pattern recognition.

Rowcliffe, J. M., Kays, R., Kranstauber, B., Carbone, C., & Jansen, P. A. (2014). Quantifying levels of animal activity using camera trap data. *Methods in Ecology and Evolution, 5*(11), 1170-1179.

Schindler, F., & Steinhage, V. (2021). Identification of animals and recognition of their actions in wildlife videos using deep learning techniques. *Ecological Informatics, 61*, 101215.

Schneider, S., Taylor, G. W., Linquist, S., & Kremer, S. C. (2019). Past, present and future approaches using computer vision for animal re-identification from camera trap data. *Methods in Ecology and Evolution, 10*(4), 461-470.

Shahinfar, S., Meek, P., & Falzon, G. (2020). “How many images do I need?” Understanding how sample size per class affects deep learning model performance metrics for balanced designs in autonomous wildlife monitoring. *Ecological Informatics, 57*, 101085.

Song, Y.-F., Zhang, Z., Shan, C., & Wang, L. (2020). *Stronger, faster and more explainable: A graph convolutional baseline for skeleton-based action recognition.* Paper presented at the proceedings of the 28th ACM international conference on multimedia.

Straat, T. v. d. (2020). *Efficient measuring of locomotion of farm animals*. Retrieved from

Tan, M., Pang, R., & Le, Q. V. (2020). *Efficientdet: Scalable and efficient object detection.* Paper presented at the Proceedings of the IEEE/CVF conference on computer vision and pattern recognition.

The DeepLabCut Team. (2021). DeepLabCut User Guide (for single animal projects). Retrieved from https://deeplabcut.github.io/DeepLabCut/docs/standardDeepLabCut_UserGuide.html

Toshev, A., & Szegedy, C. (2014). *Deeppose: Human pose estimation via deep neural networks.* Paper presented at the Proceedings of the IEEE conference on computer vision and pattern recognition.

Tuia, D., Kellenberger, B., Beery, S., Costelloe, B. R., Zuffi, S., Risse, B., . . . Burghardt, T. (2022). Perspectives in machine learning for wildlife conservation. *Nature communications, 13*(1), 1-15.

Van Etten, A., Lindenbaum, D., & Bacastow, T. M. (2018). Spacenet: A remote sensing dataset and challenge series. *arXiv preprint arXiv:1807.01232*.

Van Rossum, G. (2020). The Python Library Reference, release 3.8. 2. *Python Software Foundation, 36*.

Van Rossum, G., & Drake Jr, F. L. (1995). *Python tutorial* (Vol. 620): Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands.

Wada. (2022). Mammal's Locomotion. Retrieved from http://mammals-locomotion.com/walking.html

Waldrop, M. M. (2019). News Feature: What are the limits of deep learning? *Proceedings of the National Academy of Sciences, 116*(4), 1074-1077.

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D. A., François, R., . . . Hester, J. (2019). Welcome to the Tidyverse. *Journal of open source software, 4*(43), 1686.

Xiao, B., Wu, H., & Wei, Y. (2018). *Simple baselines for human pose estimation and tracking.* Paper presented at the Proceedings of the European conference on computer vision (ECCV).

Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2020). Dive into Deep Learning. 2020. *URL https://d2l. ai*.

Zhang, H., Cisse, M., Dauphin, Y. N., & Lopez-Paz, D. (2017). mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*.

Appendix 1: Map National Park Hoge Veluwe

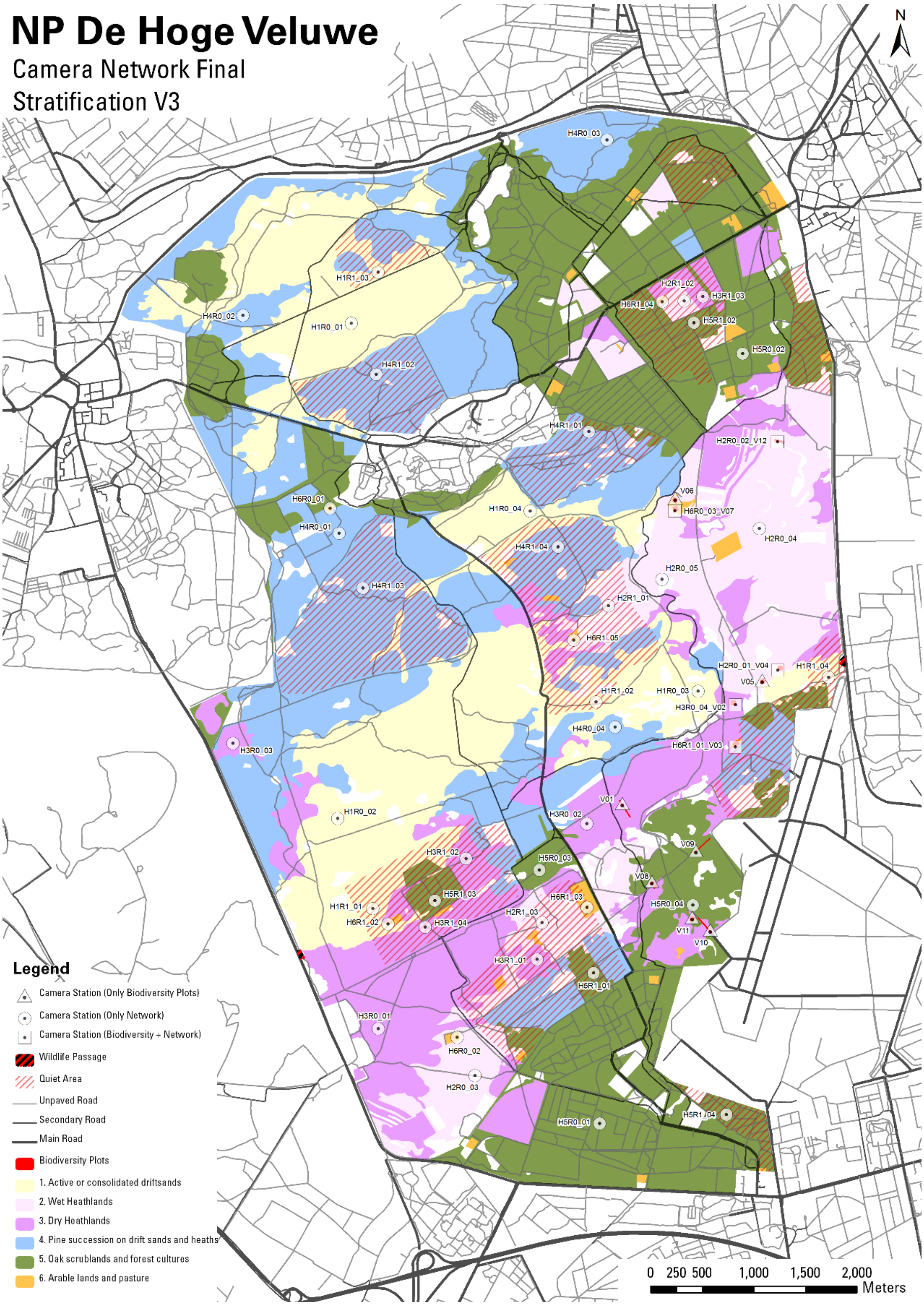
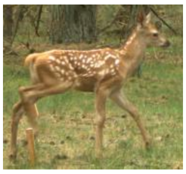
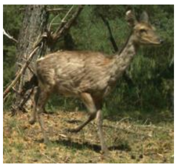
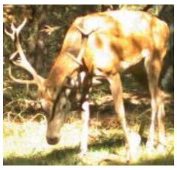
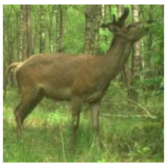
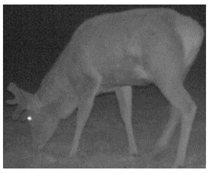
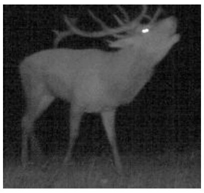
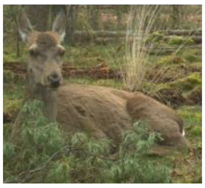
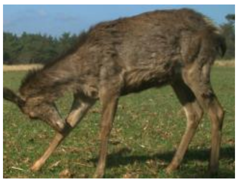
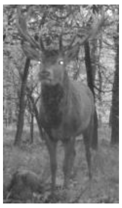
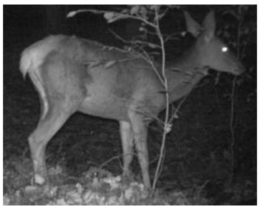
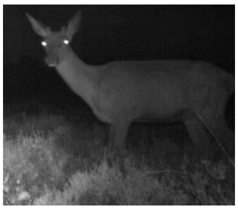
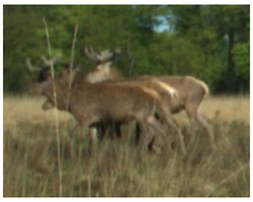


Figure 16 Detailed map of National Park Hoge Veluwe with camera locations

Appendix 2: Action annotation protocol

The Red deer (*Cervus elaphus*) action recognition criteria are shown (Table 7). The protocol for annotating wildlife actions from camera trap images is included in the reference list (Gils, 2021).

Table 7 Red deer (Cervus elaphus) action recognition criteria

Walking  Diagonal walk pattern. Front foot initiates and diagonal hind leg follows. Always two contact points, generally 3.	Running  Aerial phase with max 2 contact points visible. include trot, canter and gallop	Scanning  Mouth reaches up or down but does not touch the vegetation. Can occur either while moving or standing.	Browsing  Mouth in contact with woody vegetation. Stable leg position.
Grazing  Mouth in touch with ground or grass vegetation	Roaring  Head faced upwards and mouth opens	Sitting  Sitting or laying on the ground.	Grooming  Snout to body or leg, leg to body part or body part to object
Standing  Legs generally parallel. Not particularly tense leg position and low ear position.	Vigilance  Tense leg position in combination with pointed upward ears	Camera watching  Face and eyes point towards the camera	Interacting  Animals positioned together and react on each other

Appendix 3: Scripts and models

A. Dataset creation

- A01_data_selection.R
- A02_download_and_image_selection.R
- A03_collect_and_preprocess.R
- A04_labeling.R
- A05_create_labels_table.R
- A06_exploring_data.R

B. YOLOv5

- B01_box21_Reddeer.R
- B02_object_detection_inference.py
- B03_confusion_matrix.py

C. Pose estimation approach

- C01_terminal_DLC.py
- C02_GPU_PC_commands
- C03_create_training_dataset.py
- C04_train_and_evaluate_network.py
- C05_padding.py
- C06_analyse_time_lapse_images.py
- C07_feature_extraction.py
- C08_outliers_and_scaling.py
- C09_random_forest.py
- C10_MLP.py
- C11_dendrogram.py

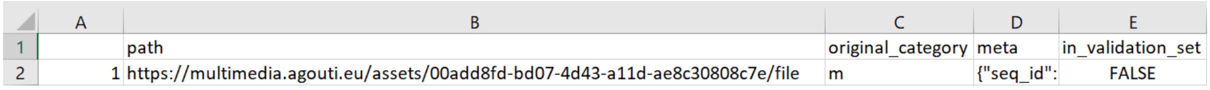
D. General

- D01_Graphs_results.py

E. Models

- E01_YOLOv5_model_parent_behaviour.pt
- E02_YOLOv5_model_behaviour.pt
- E03_PEA_RF_model_parent.joblib
- E04_PEA_RF_model.joblib

Appendix 4: BOX21 example input



	A	B	C	D	E
1		path	original_category	meta	in_validation_set
2	1	https://multimedia.agouti.eu/assets/00add8fd-bd07-4d43-a11d-ae8c30808c7e/file	m	{"seq_id":	FALSE

Figure 17 Example csv input for the YOLOV5 model on BOX21

Table 8 Description of information required per column

Column	**Description**
path	https://multimedia.agouti.eu/assets/00add8fd-bd07-4d43-a11d-ae8c30808c7e/file
original_class	m
meta	("seq_id": "46acc55c-0b90-41dd-bda7-e373b5d8651c", "depl_id": "2cc8d313-ba9c-43c6-b5d0-f254f906f2d5", "filename": "00add8fd-bd07-4d43-a11d-ae8c30808c7e.JPG") *
in_validation_set	FALSE

***Remark: change the brackets () from column meta to curly brackets**.*

Appendix 5: Train and test error graphs

Learning of the model is shown by evaluating the train and test error during training of the model. For YOLOv5 the train and test error are described by the related train and test loss (Figure 18), which describes the penalty for bad predictions, were a low loss indicates the model can predict well (Hastie et al., 2009). Results show that the loss on predicting bounding boxes is low, but prediction for the classes on the test (val) set is much higher, especially for all actions. The PEA DLC error shows how during training the average key-point pixel error evolves for the training and test dataset separately (Figure 19Figure 18). Training PEA random forest took little time, and no loss or train test error was printed

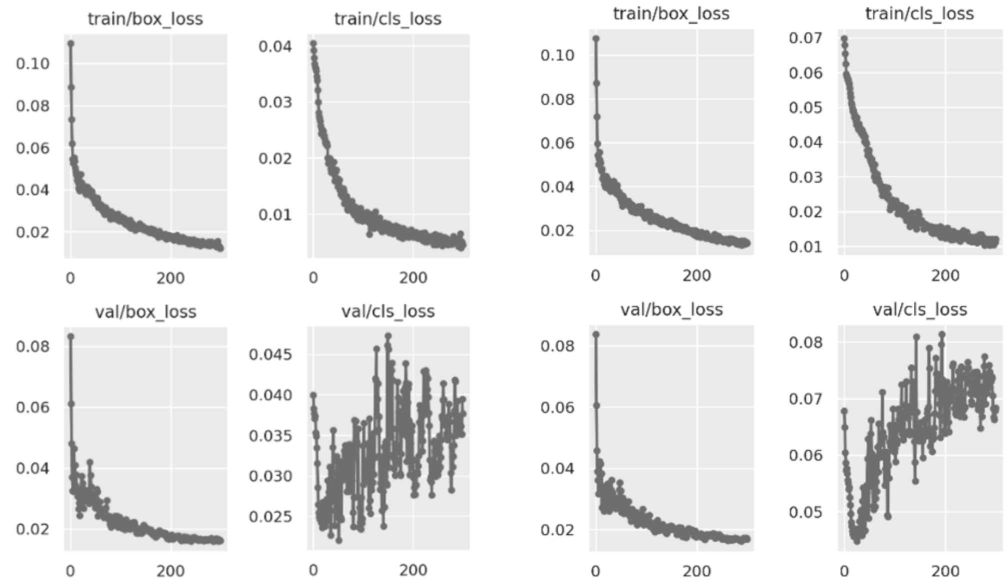


Figure 18 box loss and class loss for the parent actions (left) and all actions (right)

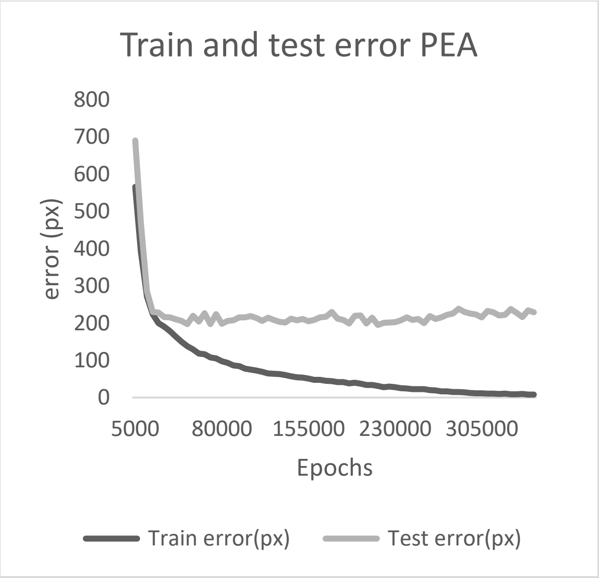


Figure 19 DLC train and test error

Appendix 6: DLC annotation protocol

The DLC annotation protocol is created to provide sufficient and accurate information about the position of the animal's key-points

To label the key-points consistently pre-decisions were made about which key-points were annotated. Based on skeleton analysis, real life camera trap images and an key-point annotated horse example (Nath et al., 2019), 18 key-points were defined covering the important Red deer (*Cervus elaphus*) joints, while still being visible from the outside (Figure 20). To obtain optimal label accuracy annotating the 506 images was divided into 25-minute parts of annotation alternated by 5 minutes break with a maximum of 3 annotation hours per day.

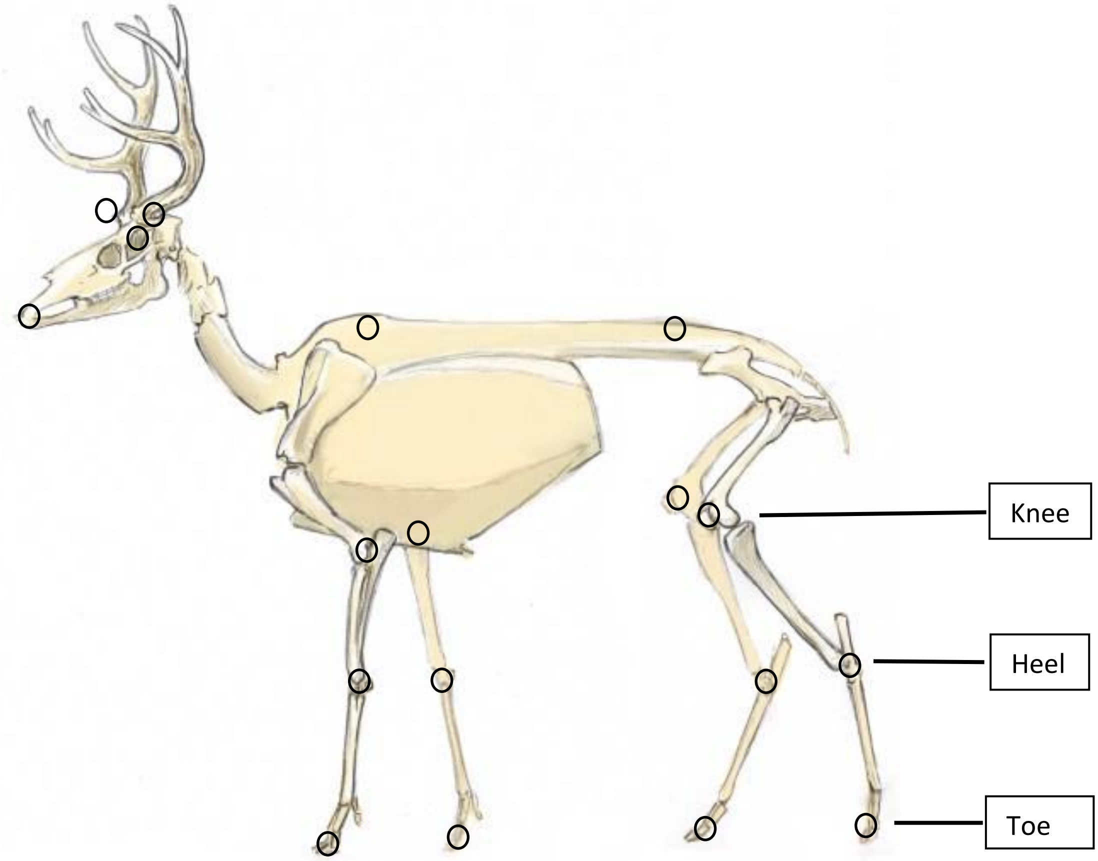


Figure 20 Key-point representation on skeleton. Image from https://johnmuirlaws.com/draw-deer-anatomy/

The key-points have been chosen relevant for detect all the 11 Red deer (*Cervus elaphus*) actions.

Not all key-points are annotated in every image. An overview of the annotation criteria is shown (Table 9). Visible key-points are annotated and occluded key-points only if occlusion is caused by non-animal objects.

Table 9 annotation criteria

Scenario	Occlusion	Annotated/Predicted
Visible key-points	no	Yes
Occluded key-points	Yes, by non-animal objects (e.g.: vegetation)	Yes
Occluded key-points	Yes, by the animal itself	No

Annotation examples show key-point annotation from the side (Figure 21) and the back (Figure 22). Key-points that were occluded by the animal's itself, were not annotated, as it was assumed these key points were hard to guess, and thus were likely to negatively influence the key-point prediction. As it was assumed that key-points occluded by non-animal objects like vegetation are often easy to guess, these key-points were annotated. To stimulate the neural network detecting contrast with the background, apparent key-points locations were chosen like the black contrasting muzzle or at the tip of the ear that contrasts with its background.

In the example below the code of a key-point example is explained.

Example: L_leg_Fw_T

1_2__3__4

1. = (L)eft or (R)ight
2. = leg or ear
3. = (F)or(w)ard or (B)ack
4. = (T)oe

To label images consistently, an order is predefined always annotating from toe to heel to knee and from left front leg to right front to right back to left back followed by the back and finally the head key-points (Table 10).

Table 10 Key-point label names

Key-point labels			
1. L_leg_Fw_T	2. L_leg_Fw_K	3. L_leg_Fw_B	4. R_leg_Fw_T
5. R_leg_Fw_K	6. R_leg_Fw_B	7. R_leg_B_T	8. R_leg_B_K
9. R_leg_B_B	10. L_leg_B_T	11. L_leg_B_K	12. L_leg_B_B
13. Low_back	14. High_back	15. Top_head	16. L_ear
17. R_ear	18. Muzzle		

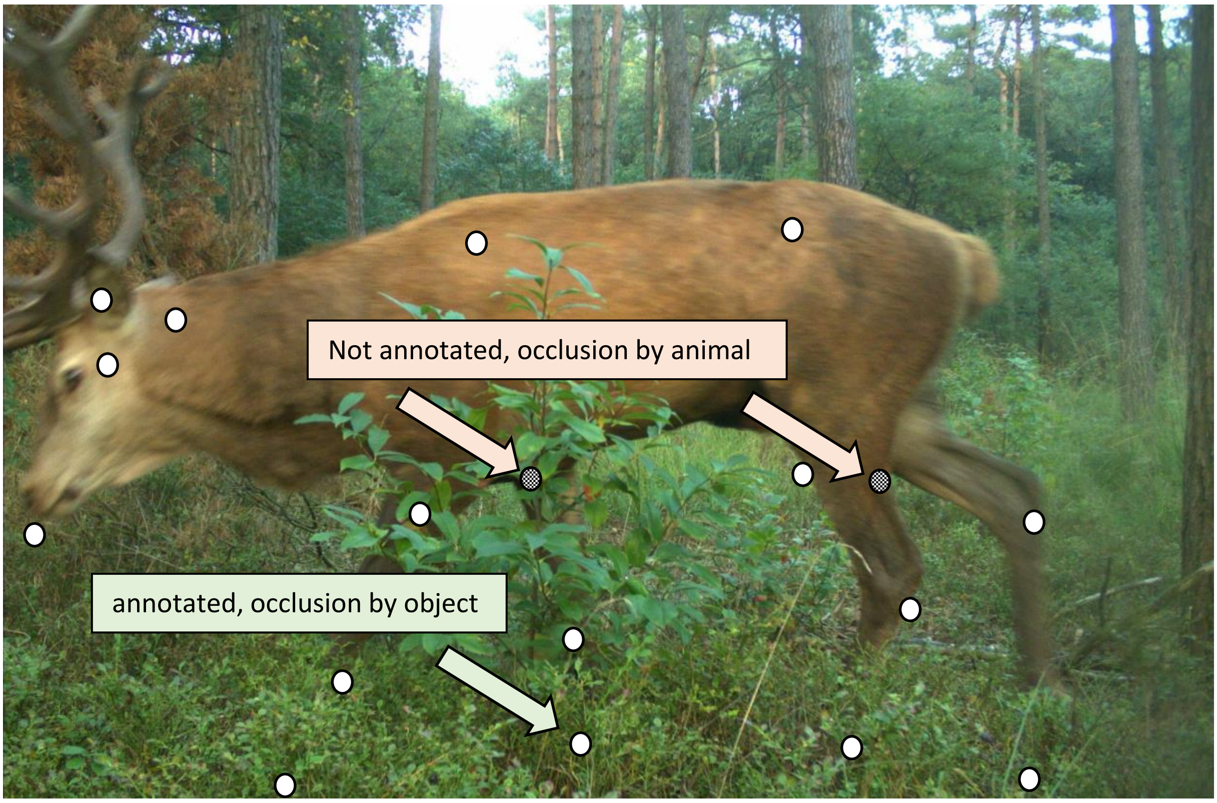

Not annotated, occlusion by animal
annotated, occlusion by object

Figure 21 annotation example sideview

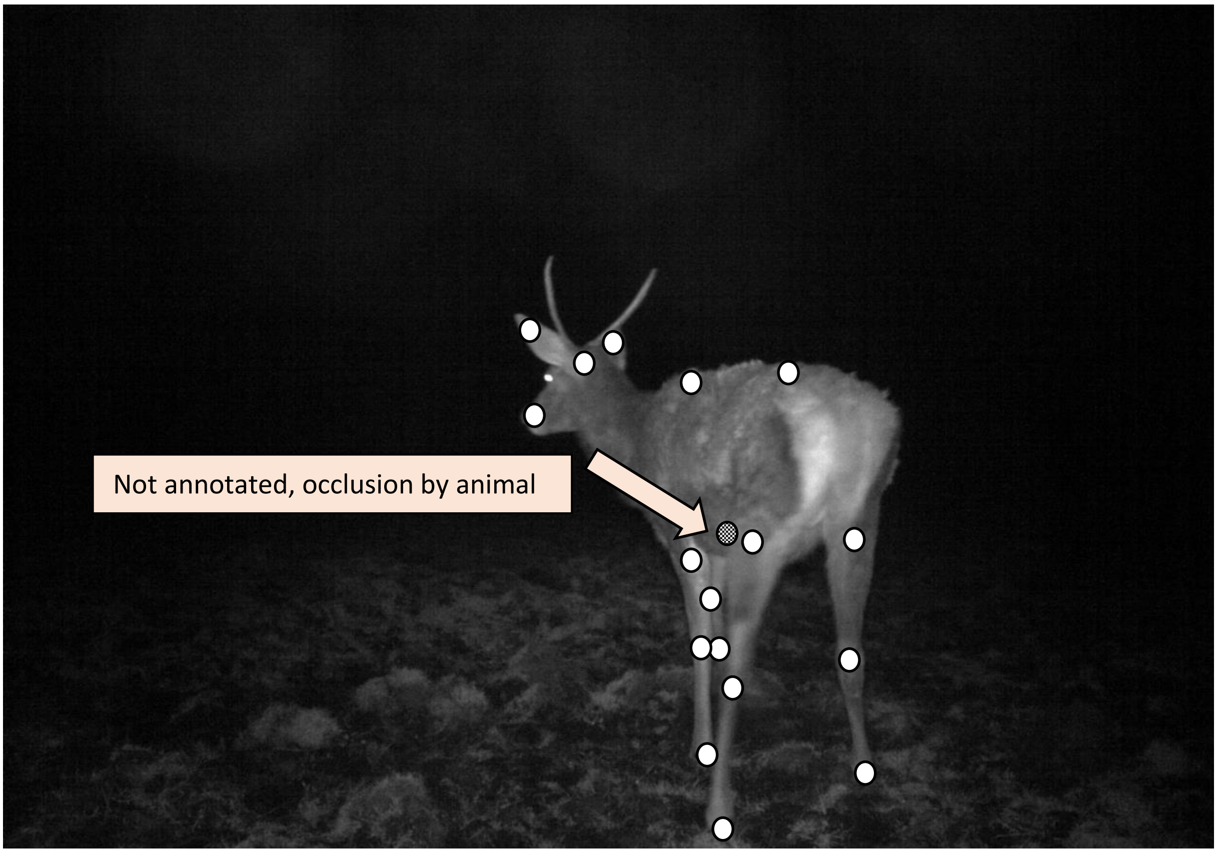


Figure 22 Annotation example behind view

Appendix 7: PEA key-point features and distribution

Table 11 Feature description of the feature extraction

Nr.	feature	abbreviation
1	shortest angle of the heel with the toe and the knee	angle_K
2	shortest angle of the knee with heel, and a hypothetical point with x value of heel and y value of knee.	angle_H
3	Vertical distance muzzle to toe, relative to vertical distance heel to toe	muzzle_toe
4	Vertical distance head to heel, relative to vertical distance knee to heel	head_heel

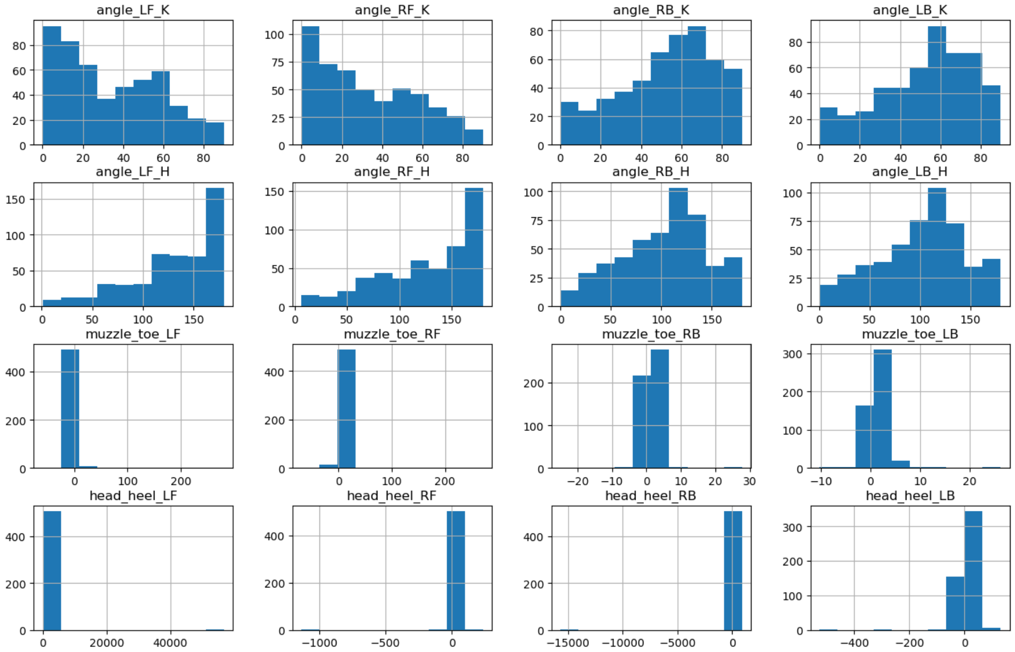


Figure 23 Distribution of all features before scaling

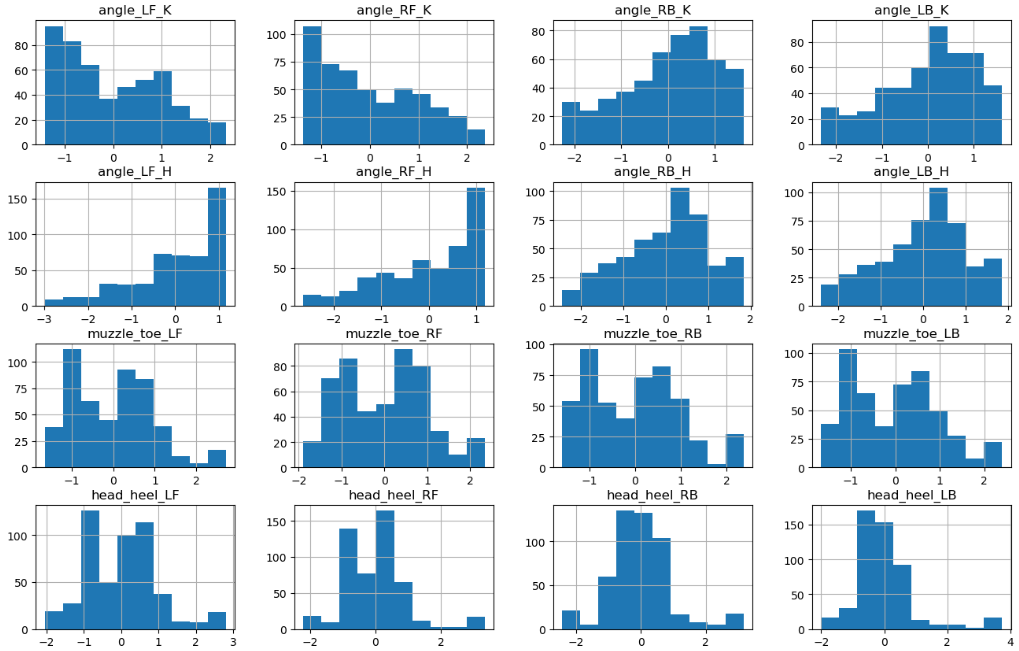


Figure 24 Distribution of all features after scaling

Appendix 8: Results

Regarding the models including all 11 actions, their accuracy (Figure 25), precision and recall (Figure 26), confusion matrixes (Figure 27) and a table of human-effort (Table 12) are shown. Finally, the dendrogram for the parent behaviours is shown (Figure 28).

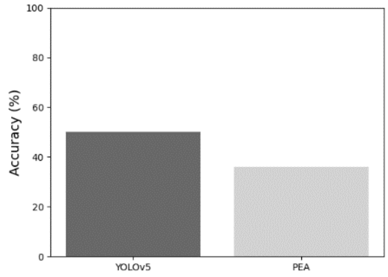


Figure 25 Overall accuracy YOLOv5 and PEA (pose estimation approach)

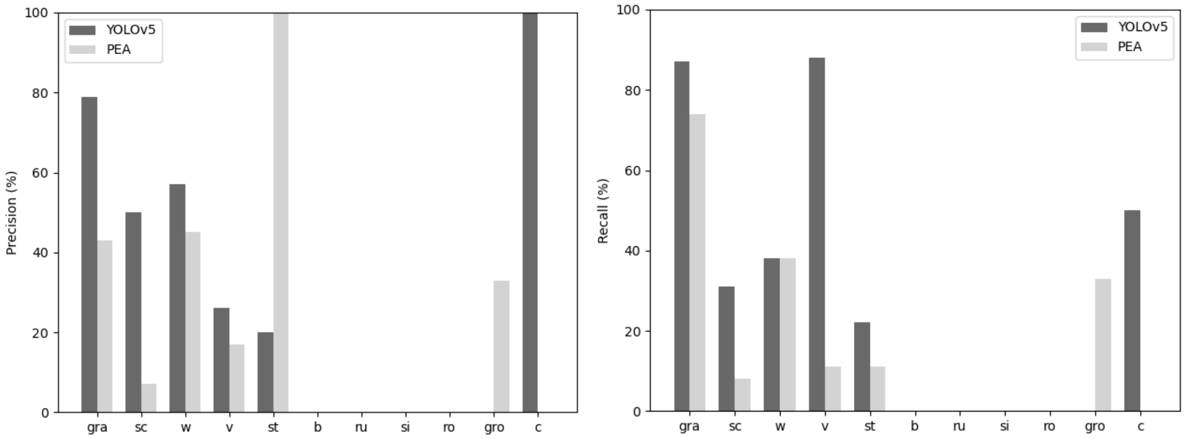


Figure 26 Precision (left) and recall (right) YOLOv5 and PEA (pose estimation approach)

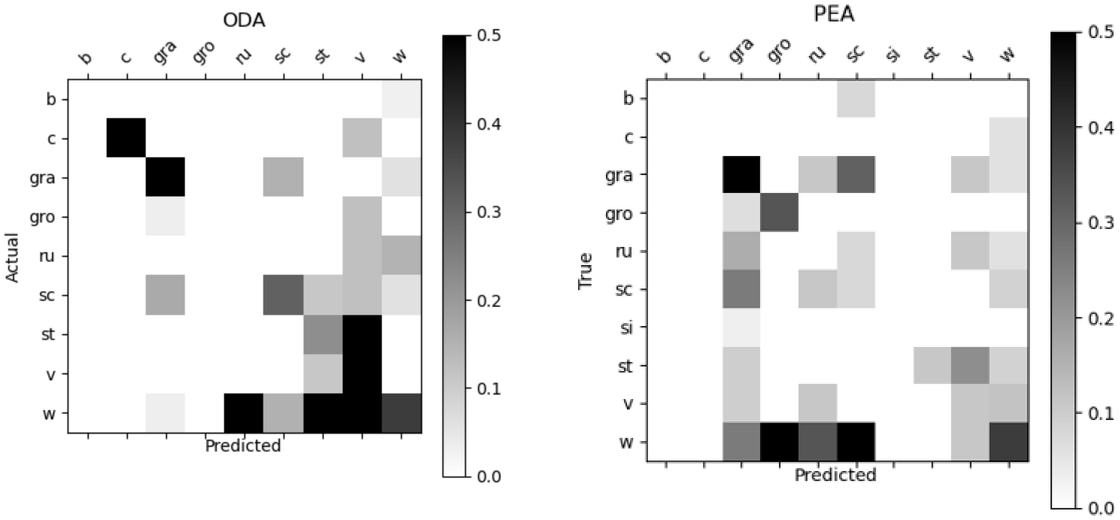


Figure 27 confusion matrix YOLOv5 (left) and PEA (pose estimation approach) (right). Predicted action (x-axis) and true labelled action (y-axis)

Table 12 Time effort YOLOv5 and PEA (pose estimation approach)

YOLOv5	**Estimated time in hours**
Data collection	22
Any Desk connection	1
Change csv to input BOX21	1
Set configuration files	1
Manually check bounding boxes	1
Train the network	1
Create confidence matrix	1
Inference	1
Total	*29*
Pose estimation approach	**Estimated building time in hours**
Data collection	22
Any Desk connection	1
Preparation	6
label images	17
Creating training dataset	5
Prepare network	5
Analyse images	3
feature extraction	3
random forest	1
Inference	1
Total	*64*

Finally, the PEA random forest dendrogram is shown evaluating the parent actions (Figure 28). The dendrogram was created by running a single decision tree based on the output of the random forest and used the Gini value as a measurement of the variation within a class, like MDI in the feature importance graph. From the top to the left side of the dendrogram we see that two-distance head to leg features (muzzle_toe_RB & muzzle_toe_Lfw). If answer to the feature criteria was yes, the group went left and if not then to the right. After two splits there was already a homogenized group with 23 foraging frames. From this dendrogram is becomes also clear that the leg angle features have not been able to create large, homogenized groups. Feature head_heel_LFw was used but did not did only separate 7 samples. Due to an error in the plotting system the number of samples were incorrect as normally the sum of the values should equal the number of samples.

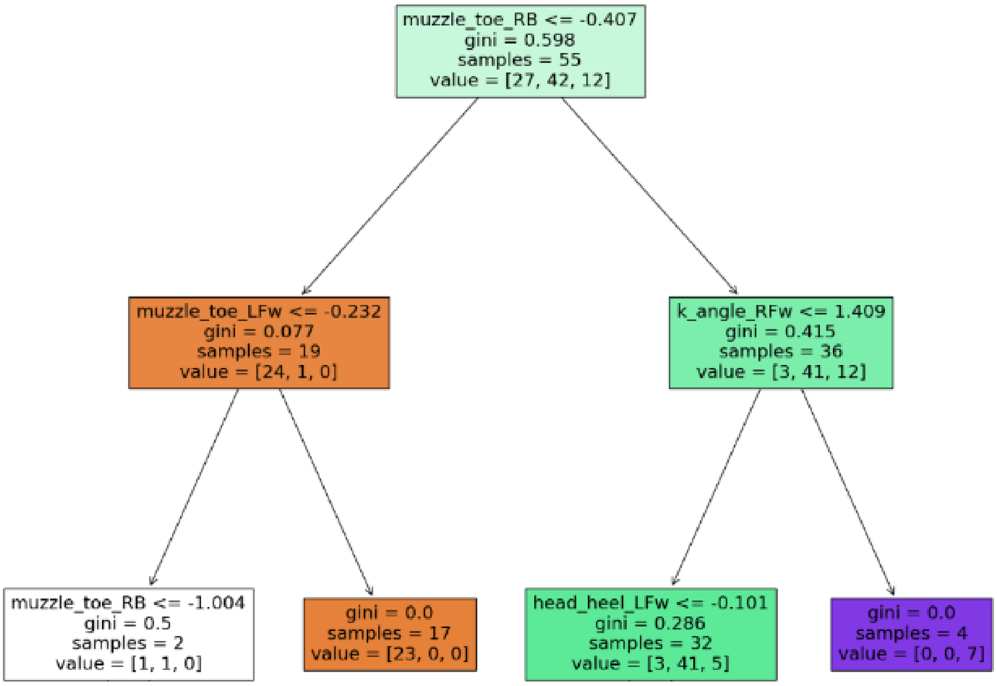


Figure 28 Decision tree based on predictions random forest. Value = [Foraging, moving, other].