

MASTER THESIS

Automated Recognition of Monkeys in Natural Habitats using Convolutional Neural Networks (CNN)

Submitted in partial fulfillment of the requirement of degree
Master of Science in Engineering

Carinthia University of Applied Sciences
Spatial Information Management

Author: Schaffhauser Lukas, BSc

Registration number: 1910362005

Supervisors: FH-Prof. Mag. Dr. Gernot Paulus, MSc MAS and
FH-Prof. Dr.-Ing. Karl-Heinrich Anders

Department of Geoinformation and Environmental
Technologies, Carinthia University of Applied Science,
Villach, Austria

Christopher D. Lippitt, PhD

Department of Geography, University of New Mexico,
Albuquerque NM, United States of America

Villach, September 2021

Statutory Declaration

I hereby declare that:

- the Master thesis has been written by myself without any external unauthorized help and that it has not been submitted to any institution to achieve an academic grading.
- I have not used sources or means without citing them in the text; any thoughts from others or literal quotations are clearly marked.
- the electronically submitted Master thesis is identical to the hard copy.
- one copy of the Master thesis is deposited and made available in the CUAS library (§ 8 Austrian Copyright Law [UrhG]).

Villach, 6th of September 2021



Schaffhauser Lukas, BSc

Abstract

The observation of animals in their natural habitats has been increasingly important over the last years. To successfully observe certain animals, the identification of those individuals has become the most important task. With the rise of Artificial Intelligence in the field of object detection, the recognition and identification of individual animals has become a lot easier. Within this thesis, a Convolutional Neural Network (CNN) was built that is capable of identifying ten Japanese Macaques living in the area of the monkey mountain in Villach, Austria.

The data used for the recognition of those ten monkeys consists of several video sequences. Those video sequences involve data from different years. To achieve the recording of different seasons of the year, an additional data acquisition took place in early spring this year. Before the data was preprocessed, the environment for the network using a You Only Look Once (YOLO) v3 architecture was built. Out of the acquired video sequences, several images for training and testing the detection of the monkeys were extracted. Those images were then split into two portions, one for training and one for testing. Both data portions were labeled within Labelbox. Out of Labelbox the labeled images were converted to the YOLO format. This data was used to train the Convolutional Neural Network. A proof of concept using a subsample of the data as well as several different trainings were performed. After the training process the network was tested using the testing data set never seen by the network. The results of the tests of the network were then evaluated calculating the Precision, the Recall, the Average Precision (AP) and the mean Average Precision (mAP). At last, the network was again tested using the acquired data from the different season.

The goal was to build a network which is capable of detecting all monkeys of interest. This goal was achieved using the precision and the performance of the network. The training within this thesis also shows that the network is capable to correctly identify the monkeys only using their body morphology. For some training runs the unbalanced amount of data for the monkeys has led to problem. The network also had issues detecting the monkeys throughout the different seasons of the year. This is caused mostly by physiological changes of the monkey's fur or the disturbance of other monkeys within the image. Those factors strongly influence the performance of such a network.

Table of Contents

1.	Introduction	4
1.1	Motivation	4
1.2	Research goals and research questions	7
1.3	Approach and Methodological Considerations.....	9
1.4	Expected Results	12
2.	Literature Overview	13
2.1	Theoretical Background	13
2.1.1	Artificial Neural Networks (ANNs)	13
2.1.2	Convolutional Neural Networks (CNNs)	15
2.1.2.1	Convolutional Layer	16
2.1.2.2	Pooling Layer	17
2.1.2.3	Fully Connected Layer and Activation Function	18
2.1.3	Image Classification	19
2.1.4	Labelbox	20
2.1.5	Crowd-participating alternative	21
2.1.6	Deep learning Frameworks	21
2.2	Technical Background	23
2.2.1	YOLO – You Only Look Once	23
2.2.2	Other object detection algorithms	24
2.3	Best Practice	25
2.3.1	CNNs used for Animal Monitoring	25
2.3.1.1	Previous work at CUAS in the field of animal recognition	25
2.3.1.2	An Animal Recognition System based on a CNN	26
2.3.2	Kibale Project – Image-based recognition of chimpanzees	27
2.3.3	Animal monitoring using different approaches	28

3.	Approach.....	29
3.1	Conceptual Workflow.....	29
3.2	Data	30
3.2.1	Requirements for the data.....	31
3.2.2	Planning the data acquisition	32
3.3	Building a Labelbox Environment.....	33
3.4	Conceptual Prototype of the CNN.....	34
3.5	Preprocessing steps.....	35
3.5.1	Preprocessing of the data	35
3.5.2	Switching from using video sequences to using single frame images.....	35
3.5.3	Building the environment of the CNN	37
3.5.3.1	CNN environment with YOLO v3.....	37
3.5.3.2	Additional software downloads	38
3.6	Proof of concept.....	39
3.7	Evaluation method	40
4.	Implementation	43
4.1	The data acquisition	43
4.2	Labelling the data	44
4.2.1	Labels	46
4.2.2	Exporting labels from Labelbox	48
4.2.3	Converting the Labelbox export to YOLO format	49
4.3	Training the CNN	52
4.4	Testing of the CNN	56
5.	Results.....	58
5.1	Proof of concept results	58
5.2	Results using all ten monkeys	61

5.2.1	Results using 15 epochs	61
5.2.2	Results using 25 epochs	65
5.2.3	Results using balanced data	67
5.3	Evaluation of the results	70
5.4	Results using the new data sets	77
6.	Discussion.....	79
7.	Conclusion & Summary.....	80
8.	Future Work.....	82
8.1	The use of more monkeys for the CNN	82
8.2	Wildlife cameras for data acquisition	82
8.3	Development of an application based on the trained network.....	83
	List of Figures.....	84
	List of Tables.....	85
	References.....	86
	Appendix A	92
	Appendix B	93

1. Introduction

1.1 Motivation

Observing animals in their natural habitat has become more and more important over the last years. Whenever animals are observed, the identification and recognition of specific focus animals is an important part of such behavioral studies (Buehler et al., 2019). Scientists, wildlife professionals and students involved in behavioral animal studies face significant challenges for tracking and monitoring their natural behavior and environment in a non-intrusive way.

With the rise of Artificial Intelligence and Machine Learning, the observation, identification and recognition of animals based on images has become a lot easier. Especially in the field of Machine Learning (ML), Convolutional Neural Networks (CNN) are applied for specific classification tasks. CNNs have been studied and implemented in many application domains, for example crowd counting (Chen et al. 2012), object detection, face attributes recognition and geo-localization (Howard et al., 2017; Girshick et al., 2014; van Gemert et al., 2014). When it comes to the recognition of animals, whether from in situ taken pictures or from wildlife cameras, the use of neural networks has become more and more important over the last years. Nowadays, most applications for the detection of animals use Convolutional Neural Networks because with the use of convolutions the pictures are scaled down so that differences can be detected way easier (Pavliček et al., 2018).

Previous work at Carinthia University of Applied Sciences (CUAS) developed a first proof of concept in terms of suitability of a Convolutional Neural Network for the automatic identification of individual monkeys. This is currently related with a lot of manual work for a small sample size, especially when training and adopting the model (Yang, 2020). The workflow proposed by Yang (2020) will be critically reflected and adopted ranging from initial image data capture to process automation to optimize the performance of the CNN. The research area of this project is the Affenberg in Landskron, Villach. It is associated with the *Austrian Research Center for Primatology* (ARCP) and is the home of 162 Japanese Macaques. Furthermore, this new approach will also be conceptually tested on chimpanzee image data from digital photo traps from the Kibale project in Uganda (Emery Thompson et al., in press). Chimpanzees (*Pan troglodytes*) have been studied at a number of long-term research sites across equatorial Africa for several decades (Goodall 1986; Nishida 1968; Wrangham 1975;

Boesch et al., 2019). During this time, attempts to quantify and qualify their ranging patterns have been hampered by factors inherent to their behavior and habitat, and the technology available for tracking them. The advent of VHF and GPS technology, which has been central to the explosion of tracking data generated in other species, has had its utility attenuated in regard to chimpanzees, who possess the strength and dexterity to remove collars (Humble et al., 2011) and rely upon a careful relationship of habituation with researchers that would be severely damaged by collaring attempts (Lonsdorf et al., 2014). Modern tracking practices therefore still rely upon direct (thought distanced) human observations, made by a team of research assistants who attempt to follow one or more chimpanzees as they range throughout the day (Emery Thompson et al., *in press*). The consistency and coverage of these observations are routinely threatened by the limits of human ability to navigate the dense understory, swamps, and steep inclines characteristic of most chimpanzee habitat (Chapman and Wrangham, 1993). Dangers including highly venomous snakes and territorial forest elephants also pose regular risks to tracking efforts. Conversely, threats originating from human researchers toward the chimpanzees themselves can limit data collection; the current Covid-19 pandemic has halted virtually all researcher presence within protected chimpanzee habitats for the last nine months (Emery Thompson, *pers comm*) because, as our closest living relative, chimpanzees are vulnerable to transmission of zoonotic disease (Kaur et al., 2008; Negrey et al., 2019).

Even when humans are successful at overcoming these obstacles, characteristics of chimpanzee behavior often limit even the most thorough tracking efforts. The fission-fusion nature of chimpanzee sociality causes groups to split and merge at an unpredictable rate throughout the day (Boesch and Lehmann, 2004), forcing human observers to necessarily abandon some members of their quarry. Furthermore, not all chimpanzees are equally habituated to observer presence. Recent female immigrants and mothers with new offspring are notoriously “shy”, and their ranging patterns go largely unrecorded. Occasionally, individuals will separate from the group and range far from traditional territory (and observer presence) for months at a time. These movements, too, remain a mystery.

Despite the difficulties associated with tracking chimpanzees, the pursuit of knowledge about their ranging patterns is a vital endeavor to both academic research and their continued survival. The IUCN Red List has identified chimpanzees as “Endangered” and “Critically Endangered” due to habitat loss, habitat fragmentation and increasing human-wildlife

conflicts across their range (Plumptre, 2010). Our knowledge about the relationship between chimpanzee ranging patterns and anthropogenic pressures is a key component for the creation of impactful conservation and management decisions and effective allocation of resources like snare-removal teams (Heinicke et al., 2019). Additionally, there remain many unanswered questions about the nature of chimpanzee behavior, biology, and ecology that can only be answered by ranging datasets with the kind of full and unbiased coverage provided by infrastructure such as a network of persistently monitoring digital camera traps. How do new mothers shift their ranging patterns to meet the needs of their offspring? What kinds of spatial interactions occur between recent immigrants? Where do the “lone rangers” go when they abandon the group, and why? Chimpanzee research is at a critical juncture in history; population decline, and the threat of extinction are growing, but so are the opportunities shepherded in by big data and the technology revolution. How we take advantage of these opportunities will no doubt influence the future of the chimpanzee.

The problem still occurring during the use of these networks is the huge amount of time the training of such a network takes. Since most of the training time is supervised training, the specialist has to look at every detail of every picture if the resulting from the network fits the expectations. This means that during the training process for the example of Japanese macaques, every detail of the face has to be detected and digitized by the expert (Yang, 2020). Since the training data for such an application needs a lot of pictures of the faces of the monkeys, and every picture needs to be digitized and classified by the expert, the development of the network is rather slow. With the addition of an automated training process, such networks will be trained in a faster way, without having problems in the quality parameters of the result. So, working in an automated will definitely help to develop better applications for animal recognition.

Another factor within neural networks is, that they do not detect new animals, they just look which of the known animals fits the best and the classify it that way. This is a big problem when automating such a network because if an automated recognition with the addition of new animals is not working, the network cannot be called automated (Yang, 2020). Of course, without some supervising, the network will never know for example the name of an animal of it is not already known by the system. Therefore, supervising is still necessary but should be in a minimal way such that experts only have to tell the network which animal this is after the network has recognized an animal which is not known. Due to this addition, the development

and the learning process of the network will improve over the time of usage. Moreover, this thesis focuses on is the recognition of the animals not only according to their face but also according to their body. This step is needed, especially when working with wildlife cameras. These cameras only detect the movement and then take pictures of the animals. But the chances are high that the picture of an animal will not show their face, so the recognition process is relatively hard. Going back to the Japanese macaques, taking pictures of their faces can be dangerous without knowing how these monkeys will react. Therefore, it is necessary to also detect these animals not only by their faces but also by their body. Because of this huge amount of new data and additional training process for such a detection, the need for an automated training is pointed out even more.

1.2 Research goals and research questions

The general objective of this master thesis is the development and validation of an automated process adopting an already existing Convolutional Neural Network prototype for the recognition of individual monkeys, which is based on the recognition of monkeys using physiognomic details in their face (Yang, 2020). Furthermore, the research in this project will focus on the process of automating itself, design of a training infrastructure, the identification and integration of new monkeys within the CNN architecture, the recognition of monkeys not only based on their face but also on their body morphology and the addition of identifying monkeys throughout the whole year addressing the issue of physiological changes (e.g. winter versus summer fur or change in face color during mating season). In addition to these objectives, the adoption and validation of a smartphone application prototype based on the Convolutional Neural Network framework (Yang, 2020) represent another important research goal. This application will provide a solution for real-time identification of monkeys in the field based on taking individual images. This approach will be tested and implemented at the Affenberg in Villach, Austria. The Affenberg Landskron is a monkey park partly accessible by tourists, but also a research field station of the University of Vienna, Austria (Austria Research Center for Primatology). Currently the park houses a population of 162 Japanese macaques (*Macaca fuscata*), including N=73 sexually matured females (≥ 3.5 years), N=52 sexually matured males (≥ 4.5 years) and N=37 immature individuals. Within a four-hectare enclosure the monkeys are kept under (semi-) free conditions and not hindered from any social interactions with group members.

Researchers and students have access to the population throughout the year to observe animal behavior as well as to conduct hormone and gene analysis using non-invasive methods. This novel approach will also be conceptually evaluated in terms of transferability and scalability to be applied on a different primate species, i.e. chimpanzee image data from digital photo traps from the Kibale project in Uganda provided by the Department of Anthropology at the University of New Mexico (Emery Thompson et al., in press; <https://kibalechimpanzees.wordpress.com/>). The objectives described in this section lead to the following research question and the following sub questions for this master thesis:

- How does the process of automation of a Convolutional Neural Network for the recognition of monkeys in their natural habitat work?
 - How does the recognition of monkeys not only based on their face but also on their body morphology and the identification throughout the whole year addressing the issue of physiological changes influence the process of recognition?
 - How does the recognition get affected by factors such as the background of the imagery, the scale of an image and the possibility of having more than one monkey in an image?

1.3 Approach and Methodological Considerations

Because of visualization purposes and the ease of understanding, the approach of this thesis with the most important steps is shown in the diagram below Figure 1. The steps shown are the main steps and also milestones which will be reached throughout the ongoing work. Every methodological consideration mentioned in the diagram is described further into this chapter.

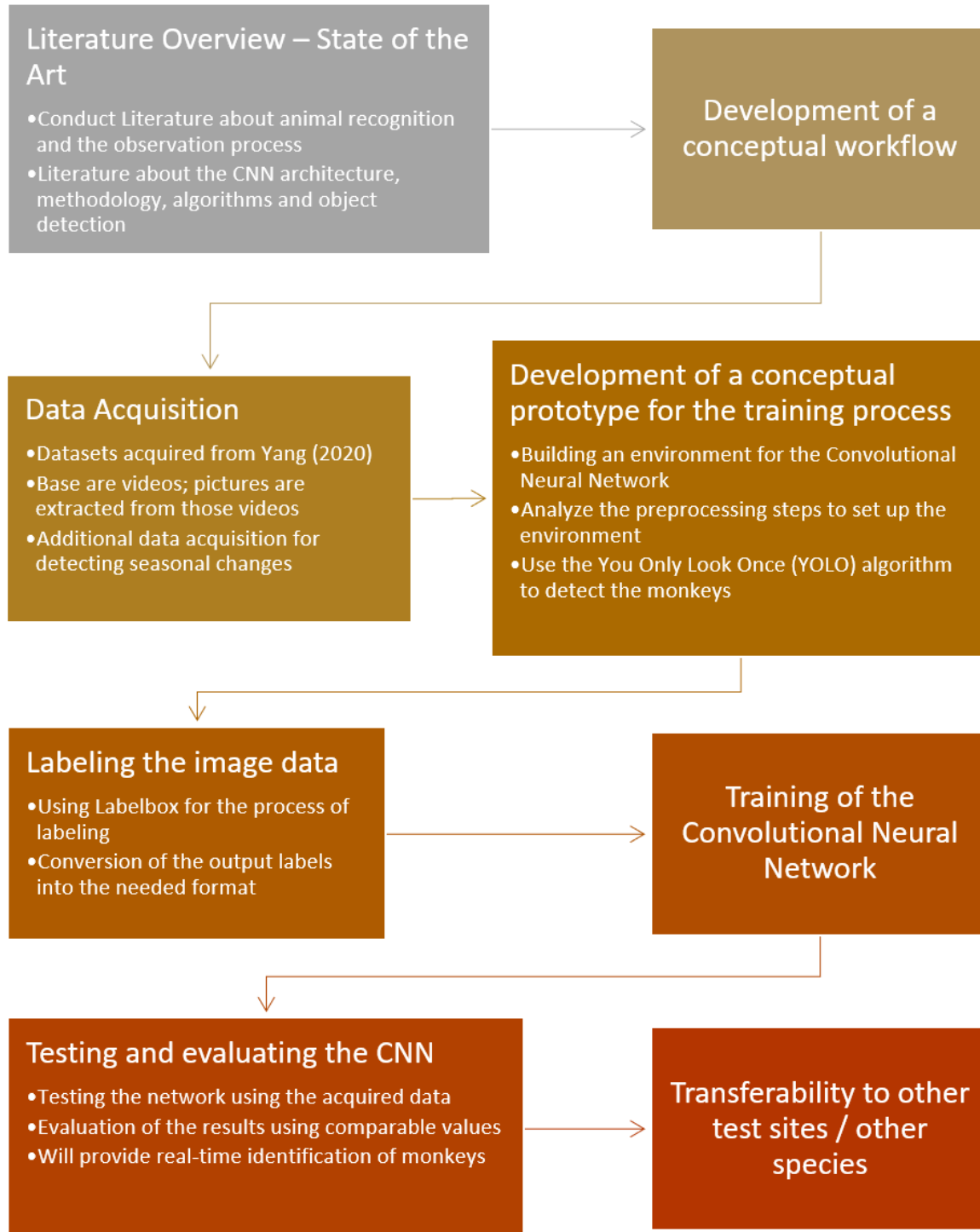


Figure 1: Workflow of the project

At the beginning of the thesis an extensive literature research will be performed. This research will give a basic knowledge of the theory. Part of understanding the methodology of the research is the structure of the Convolutional Neural Network, the requirements of the network and how the process of automation works. Additionally, information about Japanese macaques will be gathered. This is necessary to understand the different types of the fur of the macaques in the different seasons and the behavior of the animals. For the part of the development of a smartphone application, literature and knowledge on programming and designing an application will be acquired. To complete the whole literature research best practice material will be analyzed. A mind map of the described search factors and additional ones can be seen in Figure 2.

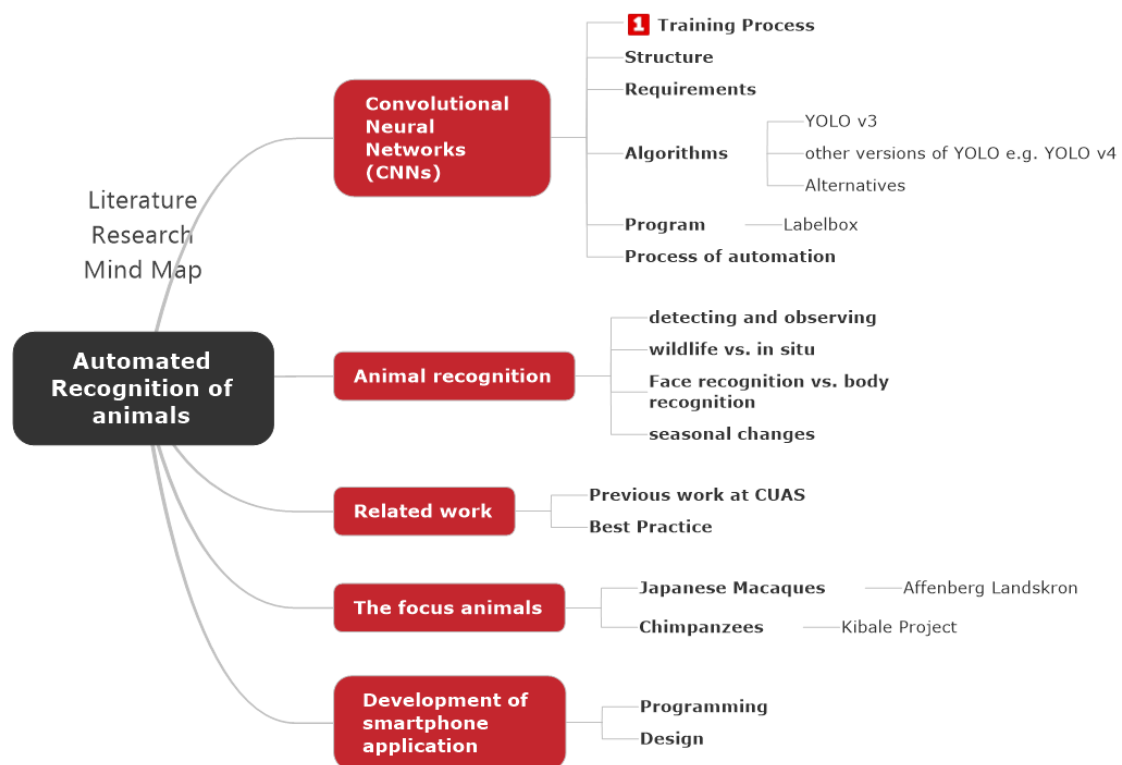


Figure 2: Literature research Mind Map

After the literature research, a conceptual workflow of the whole project will be created, and the needed data will be acquired. For the beginning, the data collected by Yang (2020) will be used. This data consists of videos of the individual monkeys. Out of these videos, the necessary images for the training process will be gathered. An additional data capturing will take place. As soon as all the necessary data is acquired, both the concept and the prototype for the training data will be created. The prototype is necessary for the success of the neural network

and the whole thesis. Furthermore, the training of the network itself will be based on how the training prototype performs. For the preparation of the input data, the platform Labelbox will be used. Labelbox is end-to-end platform to create and manage high-quality training data all in one place (Labelbox, n.d.). The already acquired pictures will be labeled in this environment. As a next step of this thesis, the design of the neural network and the training itself will take place. Although the training of such a network is really time-consuming, different trainings using different data will take place. The algorithm used for the detection of the monkeys will be YOLO (You Only Look Once). This algorithm does look at the complete image, divides it into a grid and calculates the class probability according to the created grid (Gandhi, 2018). The version of the YOLO algorithm used in this research will be YOLO v3 by Joseph Redmon and Ali Farhadi (2018). Despite newer versions of the algorithm are available, YOLO v3 will be chosen for this research. After every training both an evaluation of the networks performance as well as a look at the expectations on the results are necessary. For reaching the overall goal of automation, this process will be particularly developed to ensure the expected result. Additionally, the training process will focus on recognizing the whole monkey, not particular parts of the monkey alone. Therefore, different scales of the body parts of the monkeys will be used.

After the goal of an automated, working neural network is reached, the next step is to implement the gathered knowledge from the work with the CNN into the development of a smartphone application prototype. The focus in this work lies on the programming of the application and the implementation of factors such as usability, completeness, and the design. This application has to be tested in the field to determine if it satisfies all the expectations. Finally, the transferability and scalability of this novel approach developed for the monkey mountain in Villach, Austria will be critically evaluated and discussed based on a comprehensive specific set of requirements.

1.4 Expected Results

The expected result of this thesis is a validated prototype of a Convolutional Neural Network for the automated recognition of monkeys living in the research area. This CNN will not only detect the monkeys based on their face but also based on different parts of their body morphology. Furthermore, the CNN will be able to identify the monkeys also while dealing with seasonal changes of the fur of the monkeys. Additionally, this CNN will be used to develop a smartphone application to recognize all the monkeys by just taking a single image of them. Since all of the methods and the whole automation process can be adopted to other applications for animal detection and recognition and can also be used for different applications using automated CNNs. Especially in the field of animal tracking using wildlife cameras, the techniques used in this thesis will help to identify these animals. Therefore, the relevance of this thesis topic is expected to provide an innovative approach for the application of new technologies for behavioral animal research studies.

2. Literature Overview

To give a basic overview about the main topics of this research thesis, an extensive literature research has been made. This literature overview will explain the theory behind the used techniques and methods and is necessary for the understanding of the research reading further through the thesis. The first part needed for understanding this thesis will be the basic knowledge about Artificial Neural Networks (ANNs), which will be specialized when explaining Convolutional Neural Networks (CNNs). There will be a detailed description about the main specific parts of a CNN and how this will influence the results derived. After the information about CNNs there will be a section explaining the use of image classification, which is closely related to the use of CNNs. The section of image classification will explain why image classification is done with respect to the topic of animal detection. As a technical source of this project, different programs of interest will be introduced. To close out the overview of the necessary knowledge derived from literature, some best practice examples for the detection and recognition of animals using CNNs will be shown. These best practices will show already established techniques for the recognition of animals and will also introduce other methods.

2.1 Theoretical Background

2.1.1 Artificial Neural Networks (ANNs)

To understand the idea behind a Convolutional Neural Network (CNN), it is necessary to look at the overall understanding of Artificial Neural Networks (ANNs) before. Since there is no clear definition of an ANN, several ways of describing neural networks will be considered. Krogh (2008) describes ANNs as a type of Machine Learning (ML) which are built after the human brain itself, where the communication between the brain's neurons is done by sending electric pulses through the network of the neurons. A different definition for ANNs is given by Haykin (1999) who describes neural networks as a parallel combination of simple operating units that are able to acquire knowledge from the environmental input using a learning process and can store this gathered knowledge in the connections of the network. According to Guresen and Kayakutlu (2011) it is difficult to find one good and clear definition for Artificial Neural Networks. Nevertheless, the different ways of describing ANNs, whether from a biological or a mathematical point of view, show the complexity of such a neural network and the differences of the use of various disciplines.

Using Artificial Neural Networks for executing specific tasks brings a lot of advantages with it. Such advantages are for instance the adaptive learning of such networks, the self-organization of these networks or the ability to perform real time operations (Maind, 2014). The structure of a basic neural network can be divided into three main parts: input layers, hidden layers and output layers. When working with neural networks, the input layers are mostly multidimensional vectors, which will distribute the information to the hidden layers. These hidden layers will make decisions based on the information of the input layers and will stochastically decide how to change within itself to improve and optimize its final output (O'Shea and Nash, 2015). A simple model of a basic neural network is shown in Figure 3.

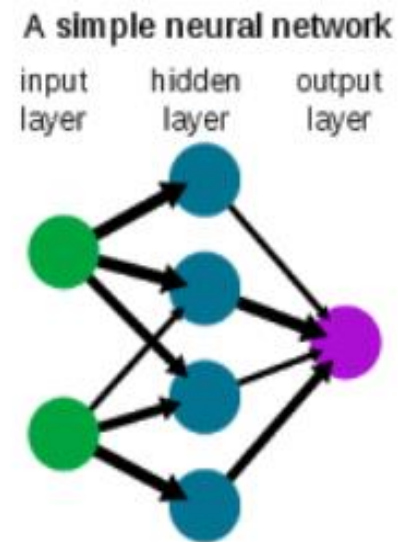


Figure 3: simple neural network; adopted from Maind, 2014

Computers with neural networks are taught to do a task by analyzing so called training data examples, which have to be previously labeled to advance these networks (Techradar, 2019). To learn how to perform such a task, neural networks cannot be programmed directly for this

task, they really need to learn the information and learn from this information. According to Maind (2014) and O'Shea and Nash (2015) there are two different ways of training an ANN, supervised and unsupervised training. These two types of training will be explained in the following list of bullets:

- **Supervised Training:** "In supervised training, both the inputs and the outputs are provided. The network then processes the inputs and compares its resulting outputs against the desired outputs. Errors are then propagated back through the system, causing the system to adjust the weights which control the network."(Maind, 2014)
- **Unsupervised Training:** "In unsupervised training, the network is provided with inputs but not with desired outputs. The system itself must then decide what features it will use to group the input data. This is often referred to as self-organization or adaption." (Maind, 2014)

In the next section the literature overview will focus on Convolutional Neural Networks specifically and on the important functionality which differs the CNN from a traditional ANN.

2.1.2 Convolutional Neural Networks (CNNs)

Since the previous section has given an overview about ANNs, this section will now focus on a specific ANN, the Convolutional Neural Network (CNN). CNNs are deep neural networks for processing data with grid patterns (Yamashita et al., 2018). According to Zhu et al. (2018), CNNs are originally designed for image analysis while Zhang et al. (2020) mentions that CNNs are widely used for performing pixelwise classification. Furthermore, CNNs are applied to natural language processing studies and computer vision (Zhu et al., 2020) which strengthens the statement from Gopika et al. (2020) who mention a wide variety of applications for CNNs in various fields.

Convolutional Neural Networks (CNNs) do not have a lot of differences compared to traditional Artificial Neural Networks (ANNs), the main difference is that CNNs are mainly used in the field of pattern recognition within images (O'Shea and Nash, 2015). As it was already mentioned describing ANNs, CNNs are also build using the three main layers of input, hidden and output layers (Zhu et al., 2020). What is special about the use of CNNs is the structure of the hidden layers. Within the hidden layers there are several layers which are the key to success for CNNs. Most of the previous work using CNNs speak about three main hidden layers: a convolutional layer, a pooling layer and fully connected layers (Yamashita et al., 2018; Zhu et al., 2018; Gopika et al., 2020; Ranjbar et al., 2020). Other works mention two additional layers for the success of a CNN: an activation function layer or nonlinearity layer (Paoletti et al., 2018; Zhu et al., 2020)

and a normalization layer (Zhu et al., 2020). Out of the two additional layers, the activation function layer is especially interesting, since it is mentioned in other works as well but not in the term of an own layer.

This nonlinearity layer is majorly mentioned within the convolutional layer and since the most used nonlinear function is the Rectified Linear Unit (ReLU), this step is mentioned when using the convolutions either with or without ReLU. A simple example of such an architecture of a CNN can be seen in Figure 4.

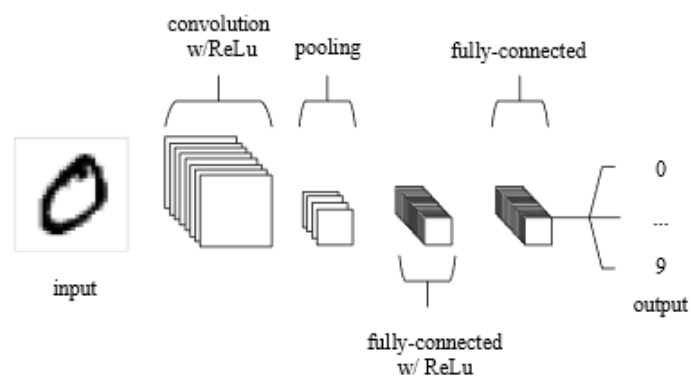


Figure 4: Simple architecture of a CNN; adopted from O'Shea and Nash, 2015

2.1.2.1 Convolutional Layer

The convolutional layer is the first layer after the input of data into a CNN. This layer is the essential layer when using CNNs, which is why these neural networks are called Convolutional neural networks. The convolutions used in a CNN compute the scalar product between local regions of the input and their weights (O'Shea and Nash, 2015). The goal of the use of convolutions is to identify from the input or previous layer and then map the appearance to a feature layer (Paoletti et al, 2018; LeCun et al., 2015). For better understanding of this step, Figure 5 shows the calculation of a convolution.

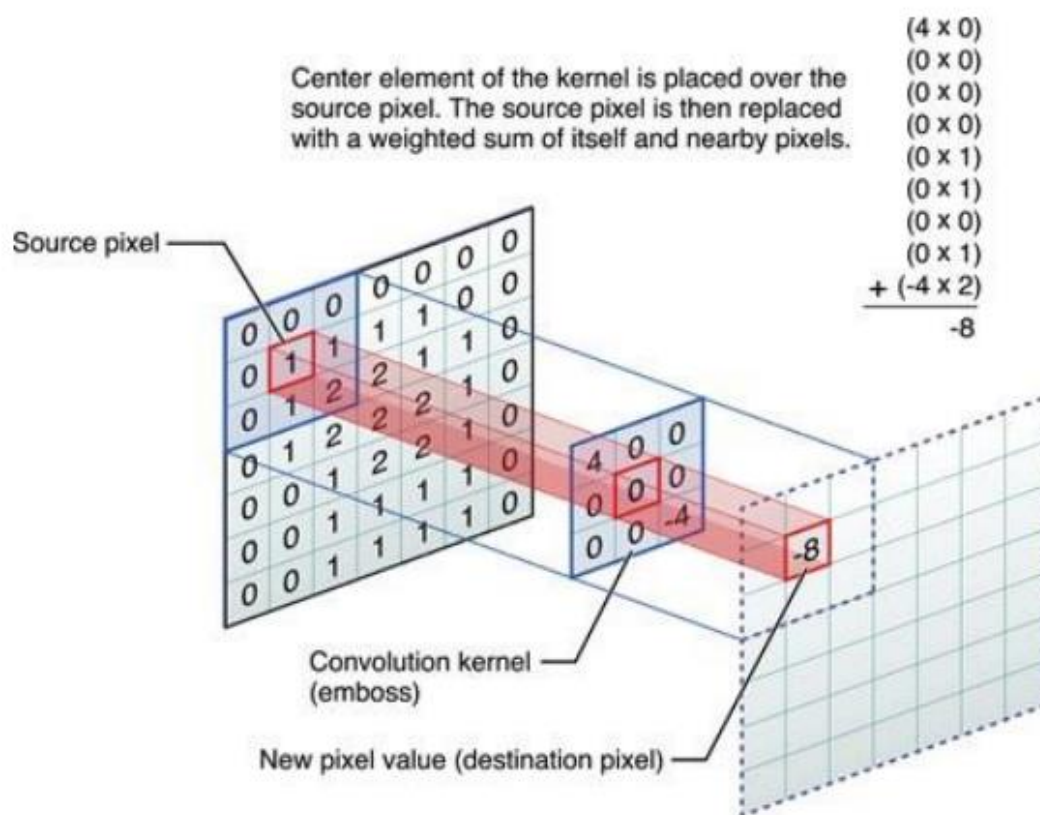


Figure 5: Calculation using a convolutional Layer; adopted from Vink, 2017

As the result of such a convolution, a feature map, which stores the information about where the feature occurs in the original input raster. A single CNN can have multiple convolutional layers and are also able to reduce the complexity of the input model through the optimization of the output significantly (O'Shea and Nash, 2015). After the use of a convolutional layer, whether at the beginning of the CNN or in the middle, a pooling layer is mostly immediately used (Ranjbar et al. 2020).

2.1.2.2 Pooling Layer

Pooling layers, also called subsampling of the CNN (Zhu et al., 2018), are used after convolutional layers. The main aim of a pooling layer is to reduce the dimensionality of the feature layer that comes out of the convolution. This is needed to reduce the complexity and the number of parameters of the model (O'Shea and Nash, 2015). Since the pooling layer is used for downscaling the dimensionality of the input, different functions have been applied for this step. The most common function used for scaling the input is the "MAX" function (O'Shea and Nash, 2015). In most cases the "MAX" function used within a pooling layer is known as max-pooling. This method works with kernels of a dimensionality of 2×2 which is applied to the input layer. Using such a pooling layer means, that out of four pixels, only the one with the highest value will be listed in the output. This results in a reduction of the dimension of the input feature layer (Zhu et al., 2018). An example of the use of a max-pooling layer can be seen in Figure 6.

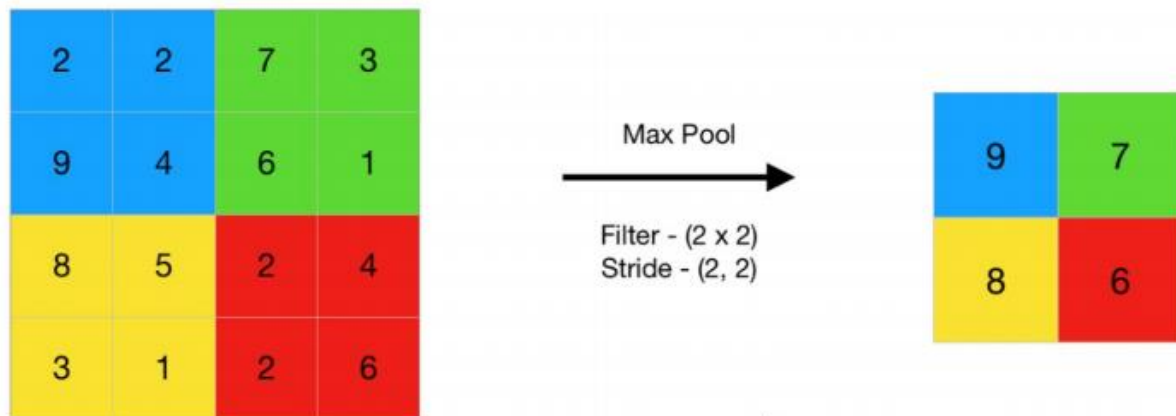


Figure 6: Max-pooling example; adopted from GeeksforGeeks, 2019

Max-pooling is not the only function used within the pooling layer. The most common alternative function is an average-pooling function which works the same way but uses, as stated in the name, the average of the input values and not the maximum value (Ranjbar et al. 2020; Zhu et al., 2018).

The use of pooling layers is necessary for the reduction of overfitting issues within the CNN and computational time of the CNN. As it is with the convolutional layers, there is not a certain number of pooling layers which delivers the best result. Pooling layers depend a lot on the input data, therefore the planning of which structure of the CNN or which algorithm is used has a big impact on the performance and results of the CNN (O'Shea and Nash, 2015).

2.1.2.3 Fully Connected Layer and Activation Function

Since two out of the three most important layers of a CNN have already been described, this section explains the need of fully connected layers within a CNN. Fully connected layers are the final layers within the structure of a CNN. Again, there can be one or more layers, which are placed after the used convolutional layers and pooling layers (Ranjbar et al., 2020). This layer is also the reason, why CNNs are often described as fully connected networks (Zhu et al., 2020).

In traditional neural networks, fully connected layers are used to reshape the feature maps coming out of the several convolutions and pooling layers into n-dimensional vectors. With these vectors, deeper and more abstract features can be extracted (Li et al., 2019) and then be used for further prediction tasks (Ranjbar et al., 2020) or as the probabilities for a class in a classification task (Yamashita et al., 2018).

Another important part of a CNN is the activation function, also mentioned as activation layer (Zhu et al., 2018;) or nonlinearity layer (Paoletti et al., 2018). This activation function is commonly applied during convolutional layers and also within the fully connected layers. The reason for the use of such an activation function is, that a neural network without an activation function would result in simple linear function, which on the one hand would be easier to solve but on the other hand has a limited complexity which results in the missing ability to learn from the data (Sharma, 2017a). Therefore, it is important to implement non-linearity into the CNN.

There are several different functions which can be applied within a Convolutional Neural Network. Sharma (2017a) mentions ten different activation functions. The best-known function for applying non-linearity to the CNN is the rectified linear unit, commonly known as ReLU (O'Shea and Nash, 2015). The formula of the ReLU function is $f(x) = \max(0, x)$. This means that all negative input values will become zero and all positive values stay the same.

The last activation function after the last fully connected layer is usually different from the ones used within the convolutions or the other fully connected layers (Yamashita et al., 2018). Depending on different tasks, different last activation functions are used. For example, if the result is a classification with multiple classes, the last activation function is needed to normalize the probabilities of the target classes (Yamashita et al., 2018; Sharma, 2017a). A typical activation function for such an example is the SoftMax function (Ranjbar et al., 2020).

2.1.3 Image Classification

Image or object classification is one of the most famous problems when dealing with image processing. Image recognition is used to detect what an input image consists of and classify that result in an output according to predefined classes (Trnovszky et al., 2017). There exist numerous methods used for solving such problems. Examples for such methods are Decision Trees, Random Forests, Support Vector Machines (SVM), Artificial Neural Networks (ANNs) (Alharbi et al., 2019), and especially Convolutional Neural Networks (CNNs) (Zhang et al., 2020; O'Shea and Nash, 2015). There is no clear statement for which method should be used for certain task, every method has its preferences and can be the best fitting for a specific problem. Nevertheless, Convolutional Neural Networks can be seen as the most consistent method for image classification (Trnovszky et al., 2017).

When trying to identify and classify specific objects using a set of images, it is important to have multiple images of the same object with different views on the object (Vasuki and Govindaraju, 2017). While talking about image classification for detecting and recognizing animals, several factors have to be considered which can cause problems in the performance of the used method. Two of the most serious degradation factors in animal recognition are Illumination and Occlusions (Trnovszky et al., 2017; Vasuki and Govindaraju, 2017). A typical animal recognition system can be seen in Figure 7.

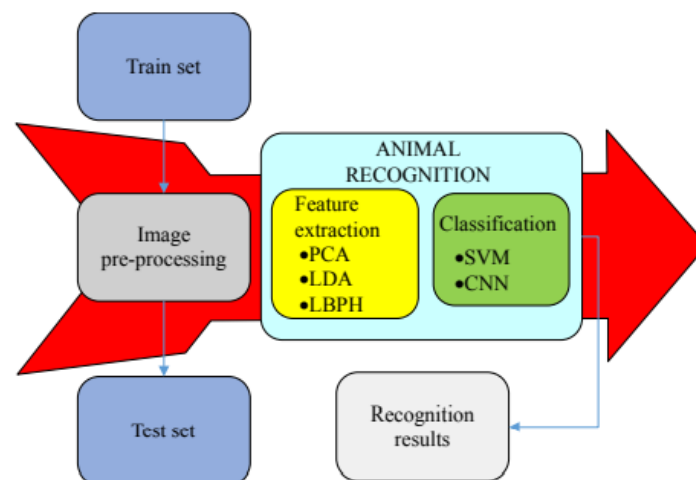


Figure 7: Animal Recognition System; adopted from Trnovszky et al., 2017

In image classification tasks an important step of preprocessing images for the use of a classifier is labeling. Labeling often is a very time-consuming and expensive work in the process of image classification, but it is a necessary step for the training process and for a correct classification (Shakya, 2020).

2.1.4 Labelbox

The first program taken into account within this project is the program Labelbox. Labelbox (2021) describes itself as an end-to-end platform to manage and create high-quality training data all in one place. Furthermore, the program can be considered as a powerful tool for image labelling, object detection and object segmentation and also image classification. The program itself delivers a lot of functions focusing on topics such as labeling and the detection of objects and individuals.

Labelbox offers three different versions for users of the program which, depending on what needed and how much money is available, contain different functionalities and have a different number of users. The three versions are a developer version, a pro version and an enterprise version. While there are no costs for the developer version, the pro version costs at least 6\$ per labeling hour. The costs for an enterprise version are depending on what the user needs and other factors. A point why this program is a good solution for the needs of this project is that Labelbox offers academic licenses, where you can get a better version of the program for free (Labelbox, 2021).

The main reason why this program will most likely be chosen for the research done in this project is that you can work with your partners within the labeling environment and can also label simultaneously, which fastens the way of working while still maintaining the needed quality. The access to this crowd labeling is with a role-based access, where the manager of the project can decide, who is working within this program and who has certain rights for the project. This is a good feature, especially when working with huge datasets where labeling would take a lot of time but with such a solution, can be split over the members of the project. Therefore, Labelbox can be considered as a standardized way to collaborate on the management of the data within the CNN.

Despite Labelbox is the program most likely used in within the scope of this project, it is necessary to look at other alternative programs and solutions, which also deal with Convolutional Neural Networks. These programs can be divided into either a crowd-participating alternative or also well-established deep learning framework solutions for working with CNNs.

2.1.5 Crowd-participating alternative

The first alternative program worth a closer look is a program for crowd-oriented projects named Zooniverse. This program describes itself as “the world’s largest and most popular platform for people-powered research” (Zooniverse, n.d.). The goal of this program is to bring together volunteers and specialists in certain fields to work together on different projects (Fortson et al., 2012; Swanson et al., 2015). What has to be taken into account is that Zooniverse works on a lot of projects where the data is gathered by specialized researchers and non-specialists are able to study those objects, such as historical images or a lot of data regarding animals in their natural habitats. The projects which are introduced by Zooniverse cover a lot of different disciplines and also various topics across science. With the help of volunteers, Zooniverse has already published a huge number of research papers where even volunteers have made significant scientific discoveries (Zooniverse, n.d.).

When looking at the different projects of Zooniverse, it can be detected that a lot of the projects deal with animals in their natural habitat. Those projects work combining the “project builder” from Zooniverse, where the researcher can supply guidelines, fulfill image classification, and define workflows that can be used for specific task volunteers in citizen science can perform. A good example for such a task to perform would be the identification of animals when using wildlife cameras (Willi et al, 2018).

2.1.6 Deep learning Frameworks

After looking at crowd-oriented software, it is also good to have a look at deep learning frameworks, which are already established in the field of working with Convolutional Neural Networks. The frameworks which will be considered within the research are three of the most known deep learning frameworks out on the market: the Keras API, PyTorch and TensorFlow.

Keras is a high-level Application Programming Interface (API) for neural networks (Keras, n.d.). The framework is written in Python and is an open-source network library specially designed for experimentations with deep neural networks. Keras can operate independently and despite the integration of Keras within TensorFlow, it can be seen as a separate library (Terra, 2021). It is built very user-friendly, simple, and readable and can be applied for especially smaller datasets. This makes Keras easy to learn and use while still maintaining productivity and minimizing the number of actions required by the user (Keras, n.d.).

While Keras is an already established framework for working with deep neural networks, PyTorch is a relatively new approach developed by the Artificial Intelligence research group of Facebook. It is also open source and since 2017 can be derived from Github (Terra, 2021). PyTorch is also known for being easy to use, flexible and good for beginners. It is written in Python and has been proven to fulfill all requirements for real-world work (Stevens et al., 2020). According to Stevens et al. (2020), PyTorch is a good starting point for data scientists, software engineers and also students with interest in Python to build deep learning neural networks in a wide range of application fields. Since the popularity of PyTorch increasing in recent times, PyTorch is becoming a good alternative for deep learning projects (Terra, 2021). The third framework worth a look is TensorFlow developed by Google. It is an end-to-end open platform for deep learning and machine learning. It is open source and is an appropriate solution for beginners and experts (TensorFlow, n.d.). It supports a lot of different platforms such as Android, iOS and Rasperry Pi. TensorFlow has a lot of different workflows for specific applications such as TensorFlow Lite for mobile embedded devices or TFX for end-to-end production (TensorFlow, n.d.). As already mentioned, TensorFlow has adopted Keras and is using it as a functional API. An easy setup from TensorFlow using Keras can be seen in Figure 8.

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
```

Figure 8: Importing Keras in TensorFlow, adopted from TensorFlow, n.d.

Since all of the previously described frameworks are good for beginners, provide a lot of examples and are good solutions for deep neural frameworks, it depends on the user's preferences which of the frameworks fits the expectations the best. The adoption of Keras from TensorFlow is still the best know example of an open-source platform for deep learning.

2.2 Technical Background

In this section of the literature overview, the technical knowledge for this thesis will be presented. The technical background of this thesis focuses on the programs which were considered during this project and will explain what program will be used, why this program fits for the research of this project and additionally an overview about other object detection algorithms, which can also be taken into account.

2.2.1 YOLO – You Only Look Once

The You Only Look Once algorithm (shortened and best known as YOLO algorithm) was developed by Joseph Redmon and Ali Farhadi in 2016. It is an algorithm that does not look at the complete image, it focuses on the parts of the image which have high probabilities of containing the needed information (Gandhi, 2018). There are different versions of the YOLO algorithm available. The first version was released in 2016 (Redmon et al., 2016), a second version called YOLO9000 was released in 2017 (Redmon and Farhadi, 2017) and the third and latest version by Redmon and Farhadi (2018) was released in 2018. There are newer versions of the algorithm available such as version 4 developed by Bochkovskiy et al. (2020). The version considered within this thesis will be version 3, because there have been good experiences using this algorithm.

The YOLO v3 algorithm consists of 53 convolutional layers. Therefore, the model is

also called Darknet-53 (Redmon and Farhadi, 2018). This narrows down an input image to the pixels with the needed information. The structure of Darknet-53 can be seen in Figure 9. As it can be seen, the is divided into several blocks. After the network architecture of YOLO v3, the most important parts of the model are the anchor boxes and the loss function (Li et al., 2019). Within the anchor boxes, the relative parameters to the anchor box are predicted. The loss

	Type	Filters	Size	Output
1x	Convolutional	32	3×3	256×256
	Convolutional	64	$3 \times 3 / 2$	128×128
	Convolutional	32	1×1	128×128
	Convolutional	64	3×3	
2x	Residual			128×128
	Convolutional	128	$3 \times 3 / 2$	64×64
	Convolutional	64	1×1	64×64
	Convolutional	128	3×3	
8x	Residual			64×64
	Convolutional	256	$3 \times 3 / 2$	32×32
	Convolutional	128	1×1	32×32
	Convolutional	256	3×3	
8x	Residual			32×32
	Convolutional	512	$3 \times 3 / 2$	16×16
	Convolutional	256	1×1	16×16
	Convolutional	512	3×3	
4x	Residual			16×16
	Convolutional	1024	$3 \times 3 / 2$	8×8
	Convolutional	512	1×1	8×8
	Convolutional	1024	3×3	
	Residual			8×8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 9: Darknet-53; adopted from Redmon and Farhadi, 2018

function is used to calculate the error between the real value and the predicted value. In YOLO v3 the loss function is the weighted sum of the localization loss, the confidence loss and the classification loss (Li et al., 2019).

To use the algorithm, it is necessary to install Darknet. According to Yang (2020), installing Darknet on Windows can be quite difficult and requires other software for the usage. The goal of this algorithm is to get a bounding box around the object and to get the class of the object (Redmon and Farhadi, 2018). In this research, the final classes of the CNN will be the names of the monkeys to identify them individually.

2.2.2 Other object detection algorithms

When dealing with object detection, there are more algorithms than only YOLO v3 available. Therefore, brief overview for alternative algorithms is necessary. The first alternatives worth a look are the region-based Convolutional Neural Network methods, namely R-CNN (Girshick et al., 2014), Fast R-CNN (Girshick, 2015) and Faster R-CNN (Ren et al., 2016). R-CNN at first is an algorithm which divides the data set into 2000 regions and continue working with them, no matter how big the input data set is (Girshick et al., 2014). Since it is the first one introduced, the problems of RCNN where still limiting the object detection since it takes huge amount of training time and is not capable of real time detection (Gandhi, 2018). Because of that, Fast R-CNN was developed. The difference to the R-CNN implementation is that Fast R-CNN is not feeding the 2000 regions to the CNN every time, instead it is fed the input image and with the use of a convolutional feature map, the regions are identified (Gandhi, 2018). Using this implementation, the biggest problems of R-CNN have been eliminated (Girshick, 2015). The difference to Faster R-CNN is, that Faster R-CNN is not using the selective search both of the other algorithms use, it lets the network learn the regions using a separate network (Gandhi, 2018). This adaption makes it usable for real time detection (Ren et al., 2016).

Another mentionable alternative is the newer version of the YOLO algorithm, YOLO v4 by Bochkovskiy et al., 2020. The difference to YOLO v3 is that it this version is faster and more accurate than version 3 by Redmon and Farhadi, 2018. The difference to the algorithms mentioned above is that with YOLO, a single CNN predicts the bounding boxes and the class predictions (Gandhi, 2018). Nevertheless, YOLO v3 was used since the algorithm has been tested a lot and promises good results for the ongoing objectives.

2.3 Best Practice

This last chapter of the literature overview introduces and describes the state of the art for the use of Convolutional Neural Networks in the field of image classification with specialization on the field of animal monitoring and animal recognition. The works described in this section have strongly influenced the research of this thesis. For each of the best practice examples the background of the research, the methods used and also the influence on this work will be described. After that, an additional method used for animal monitoring will be described.

2.3.1 CNNs used for Animal Monitoring

2.3.1.1 Previous work at CUAS in the field of animal recognition

The idea of this research thesis came up from a previously completed work at the Carinthian University of Applied Sciences (CUAS) which was done by Yang (2020). Yang has identified monkeys based on video sequences and has focused on the development of an application prototype for the identification of individual monkeys based on their face. The research done in this project ranges from the development of a prototype of a CNN, the acquisition and preprocessing of the data and mainly the development of the smartphone application prototype. The research area of his work was the Affenberg in Villach, Austria. He has chosen a subsample of the Japanese Macaques living in the research area for identifying them based on differences of their faces (Yang, 2020).

The biggest challenge Yang faced in his work was that the preprocessing of the images of the monkeys, especially the labeling part, was very time consuming. This resulted in a lot of manual work, which was all done by Yang himself. Since he used a lot of video sequences from which the images used in the ongoing process were derived, it took him way more time to label all of the input and training data. This was a big problem, especially when thinking that the same datasets and additional acquired datasets will be used within the scope of this research. Therefore, the automation of these steps was a necessary future step going out of his work.

Yang (2020) has developed a first proof of concept in terms of suitability of a Convolutional Neural Network for the automated identification of individual monkeys. Therefore, his ideas and approach will be considered and widened throughout this research. Since his project only dealt with the faces of the monkey, it was also necessary to accomplish such identification using the body morphology of the monkey and individual changes of the monkeys.

2.3.1.2 An Animal Recognition System based on a CNN

This research article from Trnovszky et al. (2017) describes an approach of the use of a Convolutional Neural Network for the classification of animal images. It compares the result derived with other, well-known image recognition methods. The methods compared to the CNN are the Principal Component Analysis (PCA), Support Vector Machines (SVM), Local Binary Patterns Histograms (LBPH) and Linear Discriminant Analysis (LDA). Their goal was to compare the mentioned methods, especially focusing on the accuracy of the approaches (Trnovszky et al., 2017). Throughout their work all of the proposed approaches are described, trained, and evaluated with the same dataset of different animals. The dataset included five different animals (wolf, bear, fox, hog and deer). Since those animals are easier to identify than the same animals of a certain species it did not take them too long to retrieve certain results. The CNN they have used for their experiments they have used images of the animals of interest, focusing on the face of the animal. They have then used two convolutional layers, the first one using a 3 x 3 kernel dimension and 16 feature maps while for the second convolution the amount of feature maps was doubled. After each convolutional layer a 2 x 2 max-pooling layer was used. At the end of the CNN, a fully connected layer with a softmax activation function was used to split the results into the five predefined classes (Trnovszky et al., 2017). The accuracy results of all five methods can be seen in Figure 10. The letters show how much of the data was used for training and how much for testing (A: 90% was used for training, B: 80% and so forth).

Ratio of test/training animal data						
	A	B	C	D	E	F
PCA (%)	85	77	72	64	62	61
LDA (%)	80	70	65	63	61	60
LBPH (%)	88	84	76	73	71	67
SVM (%)	83	74	70	68	66	64
Proposed CNN (%)	98	92	90	89	88	78

Figure 10: Method result comparison, adopted from Trnovszky et al., 2017

With those results it is proven that Convolutional Neural Networks are the best out of those approaches when trying to identify animals. Such proves of the accuracy of CNNs are the reason why they are used in a lot of classification approaches. Therefore, the use of CNNs for automated recognition is reasonable and approved.

2.3.2 Kibale Project – Image-based recognition of chimpanzees

The Kibale project is a project in Uganda (Emery Thompson et al., in press) which was established in 1987 by Dr Richard Wrangham. It has been the place of interest for several long-term studies dealing with the behavior, ecology, and physiology of chimpanzees in their natural habitat. For monitoring chimpanzees, despite modern technologies available, there is still a need for direct (through a certain distance) human observation, where teams of specialists try following specific individuals of interest (Emery Thompson et al., in press), which relies upon a close relationship between the researchers and the individual animals (Lonsdorf et al., 2014). This long-term observation project is a perfect example of the reasoning and the need for applications which detect the individual animals which will facilitate the whole observation process. There are a lot of opportunities which can be derived from the huge amount of data and the technical revolution in such research topics.

Monitoring such endangered animals comes with a lot of difficulties. The consistency of the observations is limited by human ability to not reach all of the places where chimpanzees might go, including swamps, dense understory and steep inclines which are typical for the habitat of chimpanzees (Chapman and Wrangham, 1993) and other factors such as dangerous animals e.g. venomous snakes or other limitations. Even when humans are successful at overcoming these limitations, characteristics of chimpanzee behavior often complicate tracking efforts. Using an identification approach for observing the individual chimpanzees throughout imagery or video sequences would definitely ease the process of observation. Ideas for such an application would be the use of close imagery of the chimpanzees to identify them precisely which could be derived from drone imagery or wildlife cameras (Emery Thompson et al., in press). With an approach for image-based recognition of the chimpanzees in the Kibale research area, difficulties and dangers arising during the monitoring process could be lowered or in special cases even eliminated.

2.3.3 Animal monitoring using different approaches

Buehler and his team (Buehler et al., 2019) have developed a program for the identification of individual giraffes based on images which were cropped throughout the process. The identification is based on the body morphology of the individual giraffes. The team in this project has used a different approach than a CNN, they have used Histogram of Oriented Gradients (HOG), a more traditional method for feature detection (Lowe, 1999) within a Support Vector Machine (SVM). They state that the HOG method is efficient to train and evaluate (Buehler et al., 2019). HOG detects both the boundary edges of an object and the internal texture (Dalal and Triggs, 2005) which was an important point for this research because of the specific texture of a giraffe.

The workflow provided within this work contains several steps which show some similarities to other workflows which can be seen when using Convolutional Neural Networks. The first step of the workflow was the collection of a large set of images, which are used for training and testing the model. The second step described was to create the crops of all the images, which resulted in either positive (object is within the bounding box) or negative examples, where the giraffe couldn't be seen. The third step they did was to compute the HOG descriptor which was used for the further training of the Support Vector Machine where a supervised training was used. This resulted in an image identification with a really high accuracy. Problems occurred only when the giraffe was obscured by vegetation (Buehler et al., 2019). Out of this example of animal detection using a Support Vector Machine, a lot of similarities can be derived for the workflow which will be used within this thesis. Despite stating that the program is automated, it only has been automated to some degree. Nevertheless, the approach stated in this research removes very time-consuming preprocessing steps.

3. Approach

In this chapter of the thesis, the methodological considerations of the research will be described. This will start with a conceptual workflow, where all of the following considerations are introduced and described shortly. In the further chapters, the mentioned points are described in a detailed way.

3.1 Conceptual Workflow

As a detailed overview of the further working packages of this research, a conceptual workflow was created. This workflow shows the overall steps which have to be followed for the success of the research. The mentioned steps have to be followed in the provided order and will be described in the ongoing sections and chapters of this thesis. The conceptual workflow can be seen in Figure 11.

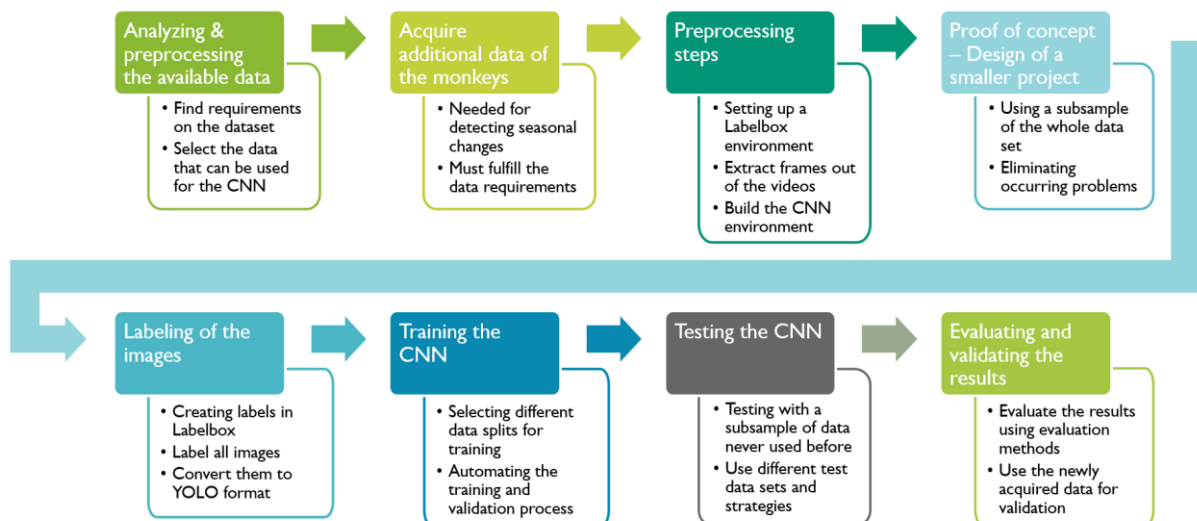


Figure 11: Conceptual Workflow of the research done in this project

This workflow will act as the guideline for the approach and the chapters after the approach. The two first points consider the already available data and the data gathered within the scope of the thesis. The next points deal with the setup for the ongoing work within the CNN. The work with the CNN can be divided into training and testing phase. Within the whole CNN, processes which can be automated will be automated. This will result in less manual work for the specialists dealing with such identification tasks.

3.2 Data

Building a Convolutional Neural Network and identifying the individual monkeys requires a lot of data. These datasets are provided by the Affenberg Villach and were acquired throughout the last years. Since there are a lot of images required for the further research steps, those datasets were acquired as video sequences. Out of those video sequences a lot of images from the individual monkeys can be derived. These datasets have already been used for training a CNN by Yang (2020), which implies that they can also be used within this thesis.

The collections of video sequences were acquired at different days and throughout the last years. Since there live 164 Japanese Macaques at the Affenberg in Villach, it is quite difficult to gather the needed data for all of the monkeys. Therefore, a subsample of the monkeys was chosen for building the CNN. When the CNN works with these selected monkeys, it is not so hard to enroll the approach used and add more of the monkeys to the CNN.

The subsample of the monkeys originally consisted of 15 Japanese Macaques, divided into ten original monkeys of interest and five additional monkeys. The ten original monkeys were acquired in April 2020 by the team of researchers at the Affenberg. The names of the ten monkeys and the time of the video sequences can be seen in Table 1. Out of the ten original monkeys, only seven of them have been found at the time of the data acquisition. Therefore, an additional data acquisition has taken place in June 2020. Within this acquisition, Yang (2020) and the researchers at the Affenberg have acquired data of five additional monkeys. These monkeys can also be seen in Table 2 with the amount of time acquired as video sequences.

Table 1: Original monkeys; adapted from Yang, 2020

The original monkeys (April 2020)	
Name of the monkey	Time of the videos acquired (in min)
Nagano	7
Zarah	4
Sandra	15
Lucky	10
Kate	7
Julius	10
Jessy	3

Table 2: Additional captured monkeys; adapted from Yang, 2020

The additional monkeys acquired (June 2020)	
Name of the monkey	Time of the videos acquired (in min)
Manfred	4
Louis	3
Conchita	10
Maya	4
Eva	12

Within these twelve mentioned monkeys, there are adult and younger ones. This will be an interesting addition to the training of the CNN because with the additional data acquisition within this thesis, we will have data available throughout a whole year which is especially interesting identifying the younger monkeys.

3.2.1 Requirements for the data

All of the gathered video sequences have to fulfill a certain number of requirements, otherwise they cannot be used for training the data. Yang (2020) has mentioned five requirements needed for the data acquisition done within his research. These can be enrolled to this thesis with additional requirements needed. The five requirements Yang (2020) mentioned are as followed:

- For the acquisition of the video sequences, a camera with a high resolution has to be used. Otherwise, the quality of the derived images is too low to extract the certain features of the monkeys.
- For each of the monkeys, there should be several video sequences with a length of at least one minute captured. This will result in enough data for the further analysis.
- To reduce the problem with occlusion, it is necessary that in a video sequence only one monkey can be seen.
- The ankles for the video sequence should range from left to right and from the bottom to the top.
- It is also stated that the orientation of the camera should be in landscape format.

Out of these requirements stated by Yang (2020), further facts which have to be considered. It is important that the distance to the individual monkeys is around one meter and can be kept throughout the whole video sequence. To accomplish this, it has to be considered that the usage of a selfie-stick or any other device to keep a distance to the monkey is necessary. Otherwise, the monkey can feel endangered which could result in problems with the monkey and could reduce the possibility of taking the needed video sequences.

Another point considered are the different scales of the different body parts of the monkey. Since the CNN in this research is trained to detect the monkey not only on the face, but it is also necessary to have data of the different body parts of the monkeys. All of these requirements on the data have to be well-thought-through before planning the additional data acquisition within this thesis.

Before the new acquisition of the monkeys can take place, it has to be stated that two of the twelve monkeys have died in the meantime. Therefore, the monkeys Julius and Lucky will not be part of the further analysis since the training of the CNN using already dead monkeys will not be beneficial for further investigations.

3.2.2 Planning the data acquisition

Since the CNN planned in this thesis will be able to detect seasonal changes of the fur of the monkeys, another data acquisition is needed. This data acquisition will take place at the end of the winter season and will look at the same monkeys of interest as the already gathered data has. At this time the monkeys will still have their seasonal fur, which will be an interesting part of the training of the CNN. Furthermore, younger monkeys like Nagano have grown a lot since the last video capture. It will be interesting, how his body morphology has changed and if the trained CNN will be smart enough to identify the monkeys despite the ongoing changes. Another challenge arising with the newly acquired data will be the background of the video sequences. Since this data acquisition is planned to take place at the end of the winter season, there will still be snow at the background. It will be interesting if and how the background will influence the identification and classification of the individual monkeys. All of the newly gathered video sequences have to fulfill the same requirements as the already acquired data. Since the data was acquired using several video sequences of the monkeys, there is the need preprocess the data. This will be done by extracting images out of the video sequences. It can be expected that not all of the extracted images will meet the requirements to be used for the

training or testing of the CNN due to illumination, occlusion or other factors which cannot be influenced, such as the monkeys cannot be found for acquiring the data or that for example the monkey cannot be captured alone.

3.3 Building a Labelbox Environment

Before starting the work with Labelbox, you first have to setup an environment within Labelbox. So, the first part is to create a project within Labelbox. When creating a new project, four steps have to be followed: the basic information of the project has to be set, the data has to be chosen (additional data can be added later on), the selection of a label editor, which is depending on which data will be used and additional settings were the percentage of the coverage can be set and the number of votes, which indicates how often each asset will be labeled, can be chosen.

After the basic settings, the overall project is created, and the creator is set as admin. As an admin of the project, the next step is to add additional team members to the project. The roles for the added members can be selected. Depending on what impact the members should have on the project, different roles such labeler, reviewer or team manager can be selected. Another important setting is the quality of the work. This is important especially when working with several labelers to include an absolute measure of quality.

Since the basic are done, there are four windows which are for importance. The overall window displays an overview of the labeling process and the quality of the training data. It can also be seen who has created which labels and how much of the workload has been done. The second window specializes in the labels itself. Here the activity and the queue of the created labels can be seen. Furthermore, also the performance of the labels and of the individual labelers can be seen. The data can be exported, either in JSON or CSV format.

Nevertheless, the most important part is the section where the labeling is done. Before labeling either an image, a video sequence, or a text, it is important to use the editor. Here the objects and classifications for the project can be set. This is an important step for the further success of the labels. After setting up the entire environment for the labeling of the image and video data, the whole labeling process can begin. The whole process of labeling with Labelbox combined with active learning from the dataset will result in a trained model with high labeling quality which can be used for a successful AI application.

3.4 Conceptual Prototype of the CNN

To identify what is needed for the successful identification of monkeys in their natural habitat, a conceptual prototype of the Convolutional Neural Network (CNN) was built. This concept will be used to identify what preferences are needed for the CNN, how the architecture of the CNN will look like, what has to be considered while building the CNN, what steps are necessary for the training and the testing and how it can be certified that the results are comparable.

At first the architecture of the CNN has to be considered. Since the already introduced YOLO algorithm will be applied, the CNN has to have a specific structure. This structure has to fulfill the reason of delivering the best possible identification of the monkeys. Therefore, the used architecture will be adapted from the YOLO structure to ensure reasonable predictions. Since this structure includes a lot of working steps, it is suggested to identify the steps and in the starting phase of the network to compute them separately. With a separate execution of the working steps, errors and mistakes within the CNN can be detected and repaired easier (Li et al., 2019).

Another consideration for the CNN is the problem with overfitting. Overfitting is when a CNN is unable to learn efficiently due to different reasons (O'Shea and Nash, 2015). This can result in a reduced ability for generalizing features within the training of the CNN and also on the prediction accuracy in the testing phase (O'Shea and Nash, 2015). To ensure that such problems will not occur it is important to build the CNN according to the available datasets which will be used, otherwise the model would not work accurately. Furthermore, precise decisions on how much of the data will be used for training have to be made before the execution of the CNN. Since the goal of this thesis is to automate the process of animal recognition, all steps possible will be simplified to make sure they will contain as less manual and time-consuming work as possible. This implies especially when labeling and preprocessing the acquired images and when training the model.

3.5 Preprocessing steps

Between the data acquisition and the planning of the CNN, some preprocessing steps have to be included. On one hand preprocessing of the used data for training, testing, and validating the CNN has to be done. On the other hand, before building the environment for the CNN, different additionally needed programs and tools have to be installed. The necessary preprocessing for both the data and the environment will be explained within the following subchapters.

3.5.1 Preprocessing of the data

The original datasets and the additionally acquired one have both been captured as video sequences. Most of these videos are available in .mp4 format, some of them have been acquired as .mts video format. This format is for videos stored in the Advanced Video Coding High Definition (AVCHD) format and is a standard format used by a lot of camcorders from Panasonic and Sony (FileInfo, n.d.). Videos of both formats can be opened within the Windows Media Player initially installed at all pcs using a Windows operating system.

The first problem arising when trying to use the videos within Labelbox is that the sequences are too big to upload. Labelbox uploads can only be at the maximum size of 256 MB, a size which limits the amount of data that can be uploaded at once. The solution to this limitation is to crop down the video sequences to smaller videos with a length of maximum twenty seconds. Using videos of this length, the data limitation of Labelbox is no problem anymore since none of the cropped videos surpasses the data limitation of 256 MB. For cropping the videos down to twenty seconds each, the program VLC media player was installed. This was necessary, since the already installed Windows Media Player is not able to crop down .mts videos.

3.5.2 Switching from using video sequences to using single frame images

As explained in the previous section, the videos have been cropped down to a size so that they can be used within Labelbox. During first tries using those video sequences, two problems resulted from the use of videos. The first problem is a minor one.

When labelling video sequences, every single frame of the video is labelled due to how Labelbox deals with video sequences. This may doesn't sound like a problem itself but for the

scope of this project to fasten the process and to automatize the Convolutional Neural Network, the use of video sequences results in using more labelled data than necessary. As an example, a twenty second video consists of approximately 500 to 550 frames. Since the acquired videos mostly focus on one monkey without really fast movement, almost one fifth of the individual frames are almost exactly the same. This results in training the CNN with around 100 images that are almost the same. When using those many images with nearly the same information provided for the CNN, the learning process of the CNN is not big due to the similarity of those labels. This means that despite the use of more data for training the CNN, the results will not get better, but the training process will take longer. Therefore, using as many frames as the video sequences can provide is not helpful for fastening the training process and automating the whole process.

The second problem arising when using video sequences is after the process of labelling. While exporting the labels from Labelbox works straight forward, accessing those labels is quite difficult. The labelling export from Labelbox for video sequences I stored using links to access the labels for each video. Those labels are secured using API keys from Labelbox. The problem is that using an API key generated in Labelbox is connected with some problems. The solution to access the labels for further analysis provided by Labelbox is to install a program called Postman, where you can access the labels. This works as claimed by Labelbox, the problem occurring is that formatting those labels so that they can be used in the ongoing analysis and as the base for the Convolutional Neural Network is quite time consuming and therefore, this method is not usable when dealing with computing time and fastening the process.

Since there were more problems occurring than advantages during first tries using video sequences, the decision was to switch to the use of single frames taken out of the video sequences. For preprocessing, again VLC media player was used to extract frames out of the used videos. At this preprocessing step, it is important to decide how many of the frames from every video should be used for training and testing.

The first step was to split up the videos into training and testing. This is necessary so that it can be assured that when testing the trained CNN, it is tested with images never seen before because using the nearly the same images for training and testing, the results do not show the capability of the network, it would only show that the network recognizes the same images that were used for training. The images were split in the following way: 70% of the data will be used for training and 30% will be used for testing. This split is a common data split when

working with CNNs. After the videos were split into the according percentages, the next working step is to extract the frames from the videos.

For extracting the frames, it depends on how many of the frames from the videos will be used. For consistency reasons, it is important to use the same frame step for all the videos so that we do not take every frame from one video and every fiftieth frame from another one. For the frame step, the first look was at the video sequences to see how much frames per second they consist of. All of the videos have 25 frames per second. After some tries using different frame steps and looking at the similarities of the individual images, to minimize the possibility of having the same scenario as while using video sequences, the decision was made to use every thirtieth frame of all the videos. This resulted in the best split of having enough data while have less similarity within the frames. Since the data is now ready for labelling, the next step is to set up the whole environment for the Convolutional Neural Network.

3.5.3 Building the environment of the CNN

After the data has been preprocessed, it is necessary to build the environment for the use of the CNN. This section describes the installations that have to be made before the calculations within the CNN can take place. There are two major steps that have to be done before starting the implementation. The first step is to install the CNN using YOLO v3, the second one is to install additional software to either fasten the whole process or for accessing the CNN.

3.5.3.1 CNN environment with YOLO v3

To build a CNN using YOLO v3, normally a lot of programming has to be done. Since the algorithm has been released in 2018, there are already existing implementations of YOLO v3 available. The CNN using YOLO v3 for this project was written by Muehlemann (2019). The whole environment with all the necessary code and a written tutorial can be accessed via GitHub.

Muehlemann recommends that the folder structure that has been used should remain the same so that no problems occur during the implementation. This object detector work with TensorFlow 2.3 and Keras 2.4. Additionally, either python 3.6 or 3.7 are recommended (Muehlemann, 2019). Within the folders of this object detection algorithm, there are five files of main interest. The first one needed according to Muehlemann (2019) is the requirements.txt file. This file contains all the necessary installations needed to later run the

code provided. The other four important files are the four python code files which are located within the numbered folders. The codes can only be used after the labelling has been finished and all the labels are in the correct format. Since the algorithm was written to use Microsoft's Visual Object Tagging Tool (VoTT), an additional data conversion from Labelbox to the format of VoTT has to be written. This conversion script will use the output from Labelbox and convert it into the same format used by the object detection algorithm.

3.5.3.2 Additional software downloads

For using the object detection algorithm provided by Muehlemann (2019), it is recommended to install certain software to fasten the process using the algorithm. The first program recommended is Anaconda Prompt. Within this program, all of the python scripts used during the algorithm will be started. Although Muehlemann (2019) is using Windows Powershell, using Anaconda prompt will help to minimize the problems while setting up the environment. The next two installations are required to fasten the whole algorithm. When training the Convolutional Neural Network, automatically the computer uses the CPU to while calculating. When dealing with object detection, the use of the GPU is recommended (Muehlemann, 2019), since this can speed up the training process, which is the step expected to take the longest. For using the GPU, two additional installations have to be made: CUDA and cuDNN. The requirement to use those additional toolkits is that a Nvidia graphic card is in use. Once this requirement is fulfilled, the correct versions of CUDA and cuDNN have to be installed. Since setting up the GPU can be quite complex, the tutorial provided by AnalyticsVidhya (n.d.) was used. This tutorial describes how to set up the GPU so that it can be used for speeding up the Convolutional Neural Network.

3.6 Proof of concept

Before starting the training using the data of all ten monkeys, the idea is to run through the whole process and the whole CNN with a smaller data sample. Therefore, three out of the ten monkeys were chosen and the whole network will be trained and tested using only the data of those three monkeys. The three chosen monkeys are Eva, Nagano and Sandra. The reason for choosing these three monkeys are that Eva and Sandra look quite similar, while Nagano is very different. This gives an opportunity in checking if the trained Convolutional Neural Network can detect even the smallest differences in the monkeys.

This smaller project, also given the name “Three Monkey Project” will not only show how good the CNN performs, but it will also have the opportunity to test different strategies and different ways to detect the individual monkeys. Since within the smaller project only the three monkeys are used for training, different training approaches and different labelling approaches can be tested way faster than when using all monkeys.

For the “Three Monkey Project”, the idea is to test out which data is necessary to correctly identify the monkeys. Therefore, three different classes for each monkey will be used. These classes will be divided into the face of the monkey, the head and the body of each individual. Since also other monkeys will appear within the images, there will also be an overall class just called monkey for the monkeys not known. This class will again be divided into three classes like the ones for the individual monkeys.

Since this project is way smaller than using the same data, different training approaches will be tested using different labels for each training. There will be a training for every class themselves, which means one training using only the face labels of the three individual monkeys and the overall monkey class for unknown monkeys, and the same for head and body of each of the labels. The fourth training will use all of the labels to see if and how this influences the performance and classification of the CNN. The fifth training that will take place is one where only images from behind will be used. This is necessary because in most of the images, the classes head and body do also contain the face and using only images from behind will show if it is necessary to have the face in the image or not.

The results of this Three Monkey Project will lead the way of how to proceed using all ten monkeys. All ideas, problems and opportunities during this smaller project will be evaluated so that during the run with all of the data, everything works straight forward, and no problems or challenges occur.

3.7 Evaluation method

Since the output of the Convolutional Neural Network using YOLO v3 are images with the corresponding bounding boxes and a table where all of this information is summarized, it is necessary to have a method to evaluate how good the network performed. To evaluate such an object detection algorithm several evaluation metrics exist (Dubey, 2020). Some of those evaluation metrics are already well established in the field of object detection and will therefore be used to evaluate the results of this project. Since the project consists of several different classes, an evaluation using the mean Average Precision (mAP) of the whole trained CNN and the Average Precision (AP) of each of the classes will be used. To use this evaluation method, three other necessary values have to be calculated: The Intersection over Union, the Precision of the class and the Recall of the class.

First the calculations for Intersection Over Union (IOU) have to be done. IOU is a measurement which compares the predicted bounding box of a detected object with the ground truth box labelled by an expert (Padilla et al., 2020). To calculate the IOU the following formula in Figure 12 has to be used.

This formula says that the intersection of the two bounding boxes has to be divided by the union of the bounding boxes. This will result in a value between 0 and 1. After the IOU has been calculated, a certain threshold has to be used to determine

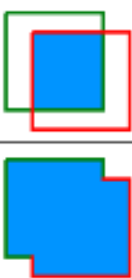
$$IOU = \frac{\text{area of overlap}}{\text{area of union}} = \frac{\text{area of overlap}}{\text{area of union}}$$


Figure 12: Intersection Over Union; adopted from Padilla et al., 2020,

which value of IOU is reasonable for the further analysis and which value is necessary to count a detection as a right detection. A common threshold to be used is 0.5 (Padilla et al., 2020). Since the IOU needs also ground truth bounding boxes labelled by an expert, not only the training data set has to be labelled but also the test data set.

After the IOU has been calculated and a certain threshold has been chosen, the results of the testing of the network have to be divided into certain classes. Those classes are True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN) which can be described as followed:

- **True Positive (TP):** This is a correct detection. The calculated IOU of the detection is higher than the chosen threshold (Padilla et al., 2020).

- **False Positive (FP):** can be described in several ways, generally it is a wrong detection or a detection with an IOU value lower than the threshold. Additionally, multiple detections of the same class within an image count as FP. Only the detection with the highest confidence will be chosen (Padilla et al., 2020).
- **False Negative (FN):** describes the case that the model did not predict the corresponding class within an image but it should have detected the object (Padilla et al., 2020).
- **True Negative (TN):** defines a correct misdetection (Padilla et al., 2020). This value is not used in object detection tasks since it would describe all the possible bounding boxes that are correctly not detected within an image which is not necessary for an object detection evaluation (Padilla et al., 2020).

Out of those four values, a confusion matrix can be made. An example of a descriptive confusion matrix can be seen in Figure 13. Since not all of the values are used for the evaluation, the further work will be done with a pseudo confusion matrix excluding the value of True Negatives.

		Predicted class	
		P	N
Actual Class	P	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

Sensitivity Recall (P) is associated with the TP cell.
 Specificity (N) is associated with the TN cell.
 Precision is associated with the FP cell.

Figure 13: Descriptive confusion matrix for evaluation; adopted from Dubey, 2020

The values of the pseudo confusion matrix are used to calculate the next two values: Precision

and Recall. The best way to understand how these values are calculated is to look at their corresponding formulas which can be seen in Figure 14.

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{\text{all detections}} \quad Recall = \frac{TP}{TP + FN} = \frac{TP}{\text{all ground truths}}$$

Figure 14: Formulas Precision and Recall; adopted from Padilla et al., 2020

For the overall values of the class these two values can be calculated using the corresponding numbers from the pseudo matrix. For the further evaluation using the Average Precision (AP), those values have to be calculated in a specified order. This is done using the confidence value of the predicted bounding boxes. At first, all the predicted bounding boxes have to be sorted according to the confidence value. After that, Precision and Recall are calculated for each predicted bounding box within the order using the confidence value (Padilla et al., 2020). This will result in changes of Precision and Recall while going through all predicted bounding boxes.

The overall value will not change. Evaluating the output data with the ordering of the confidence is used to plot a precision and recall curve. This graph will show how the relationship between precision and recall when adding more labels behaves. Furthermore, this graph will be used to determine the Average Precision for each corresponding class.

The Average Precision can be calculated using the area under the precision x recall curve. Since both values, Precision and Recall, range from 0 to 1, also the area of the graph can be maximal 1. Since this is an unlikely case for such an analysis, the value will also be between 0 and 1. To determine which values are good values, again a certain threshold has to be chosen which can describe how good the network for this certain class performed. To calculate the area under the curve, it is likely to use an interpolation for calculating the area. An example of a precision x recall curve can be seen in Figure 15.

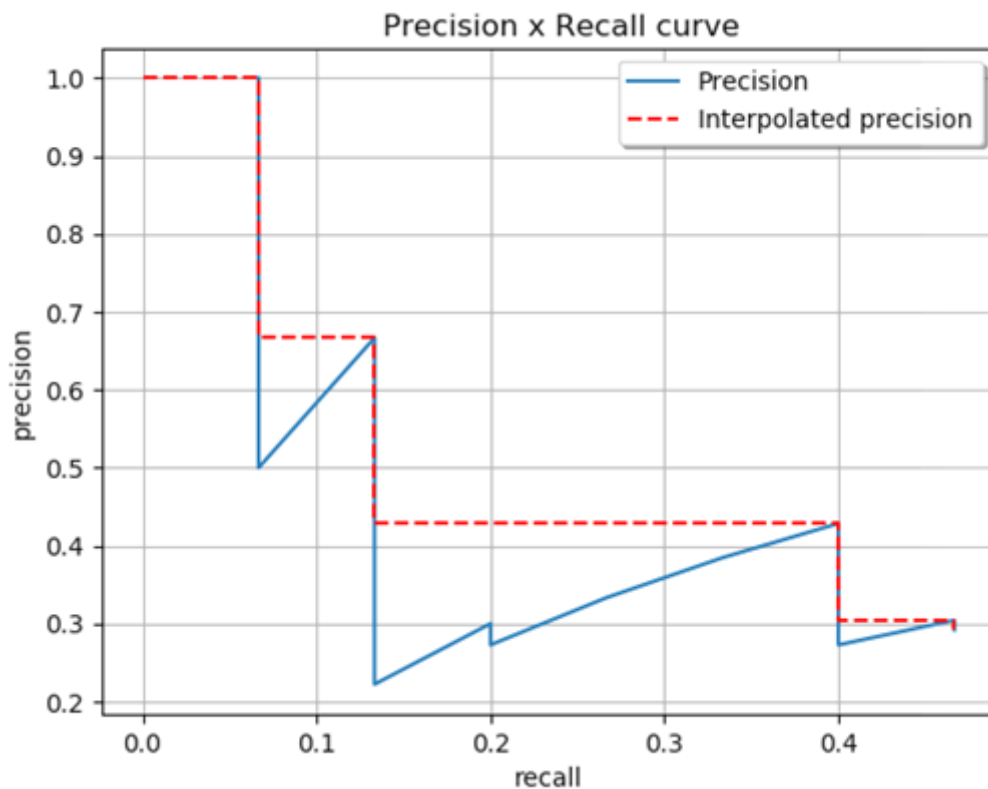


Figure 15: Example Precision x Recall curve; adopted from Padilla et al., 2020

The Average Precision can be calculated in different ways. The two most common ways of calculating the AP are the 11-point interpolation approach, which will be used later on, and applying interpolation using all points (Padilla et al., 2020). When the Average Precision has been calculated, the mean Average Precision (mAP) can be calculated using the AP of all classes from the model. This will be used to determine the performance of the trained Convolutional Neural Network using YOLO v3.

4. Implementation

This chapter of the thesis will explain the way from the previously prepared data to the final results of the Convolutional Neural Network. The steps described in this part will range from labelling the data and converting the data so that it can be used within the CNN to the training and testing phase of the CNN. Additionally, the data acquisition will be described. This data acquisition will be a necessary dataset for validating the results of the CNN. The described implementation will show the solution for the “Three-Monkey-Project”, which will then be adapted for the analysis using all ten monkeys.

4.1 The data acquisition

The newly acquired data will have an important influence on this project. Adding this data to the already acquired data last year (Yang, 2020) will not only give more possibilities during testing the data. It will also lead to the conclusions of how for example the background and the winter fur of the monkeys will influence the identification of the individuals. Henceforth, it was important that the data acquisition took place while there was still snow in the background of the monkeys.

The acquisition took place on the 8th and 9th of April. Within those two days, videos from nearly every of the needed monkeys were taken. Since it is never possible to predict whether a monkey can be found, those acquisition days were blessed since nine out of the ten monkeys were present. As an additional test for validating the CNN, three additional monkeys were recorded. Using this data as testing data for the CNN will be very interesting since it will show how the trained CNN will work using monkeys it is not trained to detect. The newly recorded monkeys are Micky, Kathi and Marco.

During the recording of the videos some difficulties occurred. Since this time of the year is also the time where most of the babies are born a lot of the monkeys are really aware that nobody comes near to the baby or the mother. Since two newly mothers are also in the group of monkeys of interest it was quite challenging to record videos from those ones. Furthermore, during the search for the monkeys of interest it was noticed that the experts working there also recognize by looking at their faces. This will be interesting since a network that can also recognize the monkeys because of different body parts would be capable of learning this information a lot faster than it would take a non-expert coming to the research area and trying

to recognize all the monkeys. For recording the videos, the SONY Alpha 7R3 and a Canon Camcorder have been used. The reason for the selection of those cameras was consistency since the original data was also collected using the same cameras. The newly acquired duration of the videos from each of the monkeys can be seen in Table 3. The yellow shaded entries are the newly acquired monkeys.

Table 3: Amount of newly acquired monkeys

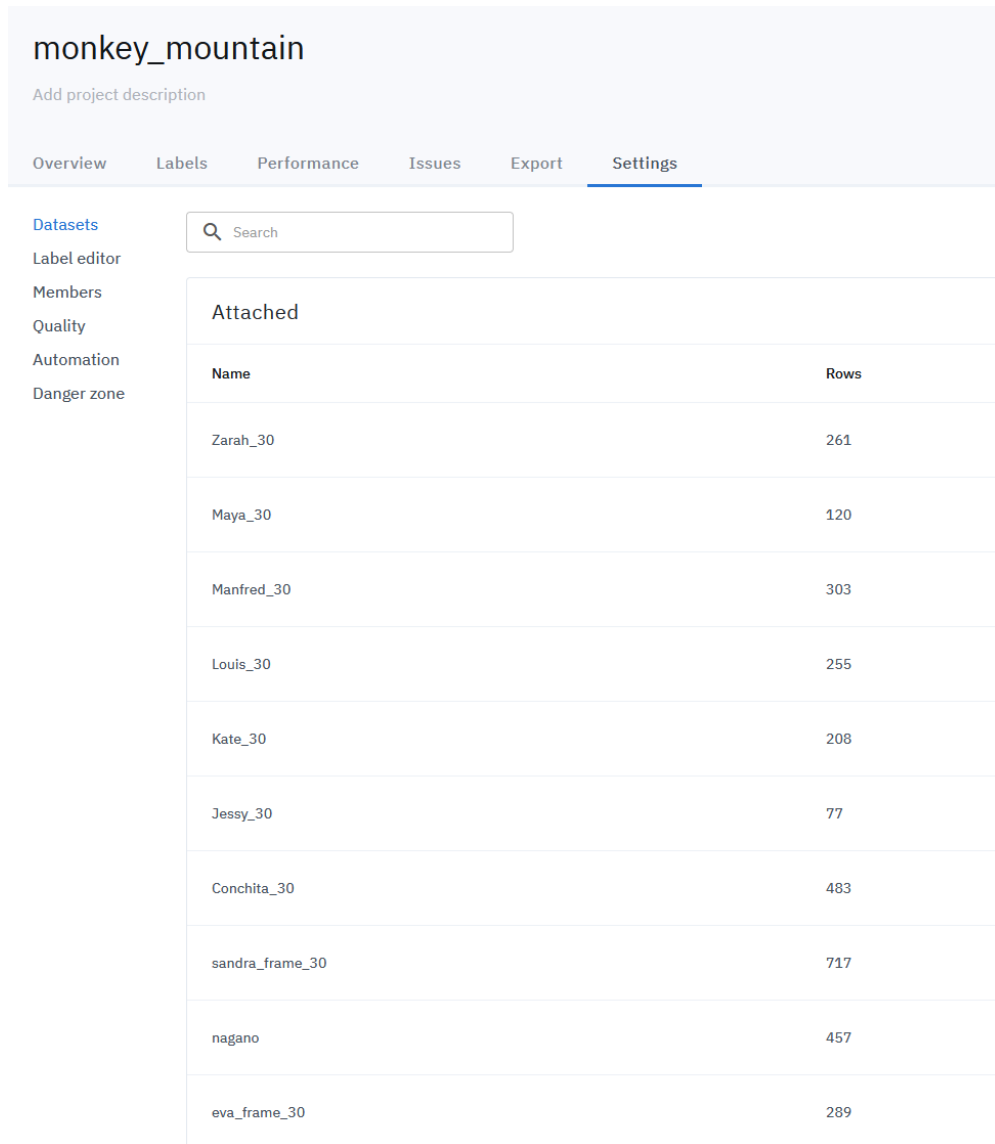
Monkey	Video Duration	Monkey	Video Duration
Conchita	5 min	Marco	9 min
Jessy	7:30 min	Micky	4 min
Kate	11 min	Nagano	3 min
Kathi	5 min	Sandra	3:30 min
Louis	5 min	Zarah	4 min
Manfred	6:30 min		

This data again has to be preprocessed to help validating the results of the CNN. Since the needed amount of images from the newly acquired videos is not as high as it was for the training process, only a few images out of every video will be taken. If using the newly recorded data will also identify the monkeys, this data set leads to a validation of the Convolutional Neural Network. It will also prove that the CNN is consistent to background changes, changes in the fur of the monkeys and also changes in the color of the face. Additionally, the newly acquired data can be used for training of a more consistent CNN and can lead to a perfect identification throughout time and different backgrounds. The supplementary recorded monkeys can then be a next step to widen the network and use more and more monkeys so that in the future all of the monkeys living in the research area can be identified using only one trained Convolutional Neural Network.

4.2 Labelling the data

As a basis for the work with the CNN, it is necessary to label all images that will be used for the training process. This step is necessary because without the information of knowing which monkey is which monkey, there cannot be a training since the network cannot learn how to identify the monkeys. As previously explained in chapter 3, the first step is to create a project

for the scope of the labelling process. The project was named “monkey_mountain”. The previously preprocessed images of the monkeys were split into the training sets of images from each monkey. Those images were then uploaded to Labelbox as separate smaller sets of images named according to the monkey’s name. The attached datasets can be seen in Figure 16.



The screenshot shows the Labelbox interface for a project named 'monkey_mountain'. The 'Settings' tab is selected, displaying a list of attached datasets. The datasets are listed in a table with columns 'Name' and 'Rows'.

Attached	
Name	Rows
Zarah_30	261
Maya_30	120
Manfred_30	303
Louis_30	255
Kate_30	208
Jessy_30	77
Conchita_30	483
sandra_frame_30	717
nagano	457
eva_frame_30	289

Figure 16: Data sets imported into Labelbox

While setting up the project, there are various settings that can be chosen from to fulfill all requirements for the project. Those settings should be selected according to how they are needed for the further analysis. The most important setting is to edit all necessary labels for the scope of the project. The labels will be shown in the following section. Other settings while setting up the project are the members of the project and their roles as well as quality settings such as benchmarks which help to set a gold standard for larger projects, consensus to cover

the number of assets and how often an asset has to be labelled and if the project will be reviewed or not. Those quality settings are mostly used while working as a team within Labelbox since they set standards which all labelers have to fulfill. It is important to choose those quality measurements before starting to label since changes during the project will influence the equality and similarity of the labels.

4.2.1 Labels

As previously described, the labels of the of the project have to be edited. This has to be done before starting with the labeling, but those labels can also be edited after some images already have been labeled. For the final training data set twenty labels have been created. Those labels are shown in Figure 17. The numbers next to the first nine labels are shortcuts so that the process of labeling can be fastened. For the prove of concept, other labels have been used. This will be shown when evaluating the results.

<input type="checkbox"/> Conchita	1	<input type="checkbox"/> Maya	
<input type="checkbox"/> Conchita_face	2	<input type="checkbox"/> Maya_face	
<input type="checkbox"/> Jessy	3	<input type="checkbox"/> Zarah	
<input type="checkbox"/> Jessy_face	4	<input type="checkbox"/> Zarah_face	
<input type="checkbox"/> Kate	5	<input type="checkbox"/> Eva	
<input type="checkbox"/> Kate_face	6	<input type="checkbox"/> Eva_face	
<input type="checkbox"/> Louis	7	<input type="checkbox"/> Nagano	
<input type="checkbox"/> Louis_face	8	<input type="checkbox"/> Nagano_face	
<input type="checkbox"/> Manfred	9	<input type="checkbox"/> Sandra	
<input type="checkbox"/> Manfred_face		<input type="checkbox"/> Sandra_face	

Figure 17: Created labels for labeling the monkeys

During editing those labels, different types for the annotations can be selected. For this project the type of annotation is a bounding box. Bounding boxes were chosen because the input format for the CNN and therefore the output format have to be the coordinates of two corners of the bounding box. Other annotation types are polygon, polyline, point entity and segmentation. It is important to know that after one of those objects has been used to label a

monkey in an image, the annotation type cannot be changed. Therefore, the selection of those types has to be done before they are used for labeling.

To start labeling the images, one image of the uploaded datasets is selected randomly and displayed in the labeling window. This indicates that knowing all the different monkeys in the images is necessary before starting to label them. For labeling the images, the corresponding annotation class has to be chosen. When the right class is chosen, a bounding box can be drawn around the body part of interest of the monkey. An example of a labeled image can be seen in Figure 18.



Figure 18: Example of a labeled image in Labelbox

Since the labels are used for the face, the head and the body of the monkey, it is likely that the labels overlap. But since all labels are considered as individual labels, there are no problems occurring when using those labels in a generated output. Furthermore, if a label is misplaced or the wrong class was chosen it can be adapted afterwards. Such changes can be the change of the corresponding class, copying or duplicating the label, resizing the corresponding bounding box or deleting the annotation. If an image was already labeled and has to be changed later in the process, the already created labels can still be changed and if all labels are deleted, Labelbox requeues the image automatically so it can be labeled again.

In some cases, images do not show the monkey of interest or are just too blurry to be used. If such a case occurs, the image can be skipped. This will not requeue the image as explained before, it will mark the image as skipped, which will also result in a different output for the specific image. If the image was accidentally skipped, it can again be requeued for the labeling process.

4.2.2 Exporting labels from Labelbox

After every image has been labeled, an export can be generated from Labelbox. This export is in JSON format and can either be downloaded directly from the export section in the project or using a Software Development Kit (SDK). The code for using the SDK is also available in the export section of the project. Such an export can be generated anytime during the ongoing project for intermediate usage but the export containing the labels from all uploaded images is only available after finishing all annotations.

Within the scope of the “Three-Monkey-Project” 1463 images were used. Therefore, the export generated by Labelbox consists of 1463 data rows. This also includes images where more than one annotation had to be done and images that have been skipped. After downloading, the export is saved as a .json file in the download folder of the computer. To open and edit such a file, Notepad++ was used. As already mentioned, the output contains of 1463 rows, each representing the annotations from one particular image. As an example, one data row can be seen in Appendix A.

Since those data rows contain a lot of information, it is important to gather the necessary parts needed for the later training of the Convolutional Neural Network using YOLO v3. Therefore, the most needed information out of the complete annotation of one image are:

- The name of the image as an id
- The amount of labels within this image
- The class for each of those labels
- The coordinates of the bounding for each of the classes
- The information if the image was skipped

This information has to be excluded from the data row. To see the differences of a data row containing all the information and a data row of a skipped image, the data row of a skipped image can also be seen in Appendix A. The selection of information limited to those five factors is to gather all the necessary information that will be needed to further continue with the work of the CNN. These gathered attributes have to be the same as the ones used by

Muehlemann (2019). In the next section it will be explained which parts of the data row have stored the necessary information, how to access them using python code and how to convert the exported data rows generated by Labelbox so that they can be used for the further analysis using the CNN.

4.2.3 Converting the Labelbox export to YOLO format

Since the further code for training and testing written by Muehlemann(2019) is designed for the work with Microsoft's Visual Object Tagging Tool (VoTT), it is also written so that the output from VoTT can directly be used for the further analysis. Because rewriting the code so that it fits the format of the Labelbox export takes quite some time, it was faster to write a code snippet that converts the export format from Labelbox to the format from VoTT. As a starting point for this code, it is necessary to know to which format or sequence of attributes the export from Labelbox has to be converted. The attributes have to be in the following order: first the name of the image, second the coordinates of the bounding box and at last the name of the label itself.

To find the information about the name of the image, which is giving the output the identification to refer to the different bounding boxes within the image, we have to look at one of the last parts of the Labelbox output called "External ID". In this part of the data row the only information stored is the name of the image with the corresponding ending of the data format. The other IDs in the data row which are generated by Labelbox do not help with the further analysis since those are only for the use within Labelbox to access the correct images. The second source of information to look at each data row from Labelbox is the section called "Label". Within this section an objects array is located which has all the other necessary information stored. To distinguish between the different classes within an image, it is important to mention that each object within this label section refers to a different annotation within the image. Knowing this, counting the objects within a data row leads to the answer of how many different annotations within the image are.

When looking at each of the objects individually, it can be detected that each object has its own title. This title refers to the class of the label and is also the information of the label needed for the ongoing analysis. Additionally, each of the individual objects has its own part within the array called "bbox". Within this small array the four coordinates of the bounding box of the label are stored. These coordinates are stored as "left", "top", "height" and "width".

Since VoTT stores the information for the bounding box in a different way, those coordinates have to be converted into the coordinates using the minimum of x and y for the upper left corner of the bounding box and the maximum of x and y for the lower right corner of the bounding box. For calculating those new coordinates, it is important to gather the information of which original coordinate refers to which converted coordinate. The calculations for each of the four coordinates are as follows:

- X minimum = "left"
- Y minimum = "top"
- X maximum = "left" + "width"
- Y maximum = "top " + "height"

At last, the focus lies on detecting skipped images, because this information will not be needed for the further analysis and therefore, those images will be excluded from the output conversion. To find skipped images, there are two ways. The first way is that the "Label" section of the data row, where most of the needed information is stored, is empty. This implies that checking for the emptiness of the "Label" section can easily be included into the code since the information in this section is needed anyway. The second way of checking if an image was skipped has occurred during the ongoing work within the project and is not included in older exports from Labelbox. With an update of the export, a Boolean operator at the end of the data row has been added which indicates if an image was skipped or not. Since this was added after the first labels have been converted, the code remained the same because the other method is still working. To gather all of this information in a short time, a code for the conversion from Labelbox to the VoTT format was written. This code can be seen in Figure 19. Using this code, the export from Labelbox is converted as explained before and will be saved as a.csv file. This is necessary to convert the generated .csv file to an input that YOLO v3 can work with. Within the code it can be seen how the different attributes of each individual image were selected and how the coordinates of the bounding box were calculated. This code works using two for loops. The iterations for those loops, called "n" and "m", guide through the loops. The first one (n) is iterating through the data rows and selecting each image after another. The other iteration (m) is going through the amount of annotations within an image. The last row after the for loops saves all the gathered information into a .csv file. Because of the better readability in Figure 19, the last line was split into two rows. For using the code, these two rows have to be combined again.

```

1 import pandas as pd
2 import json
3 import csv
4 import re
5
6
7 with open(r'C:\Users\Benutzer1\Desktop\2nd_try\TrainYourOwnYOLO\export-2021-08-01T17 00 31.545Z.json') as f:
8     data = json.load(f)
9
10
11 df = pd.DataFrame(data)
12 lb=df['Label']
13 im=df['External ID']
14
15 df_annotations = pd.DataFrame(columns= ['image', 'xmin', 'ymin', 'xmax', 'ymax', 'label'])
16
17 for n in range(len(lb)):
18     prov = lb[n]
19     if not prov:
20         n+1
21     else:
22         for m in range(len(lb[n]['objects'])):
23             lb_co = lb[n]['objects'][m]['bbox']
24             im_name = im[n]
25             xmin = lb_co['left']
26             ymin = lb_co['top']
27             xmax = lb_co['width'] + xmin
28             ymax = lb_co['height'] + ymin
29             labl = lb[n]['objects'][m]['title']
30
31
32     df_annotations = df_annotations.append(
33         {
34             'image': im_name,
35             'xmin': xmin,
36             'ymin': ymin,
37             'xmax': xmax,
38             'ymax': ymax,
39             'label': labl
40         }, ignore_index=True
41     )
42
43
44 df_annotations.to_csv(r'C:\Users\Benutzer1\Desktop\2nd_try\TrainYourOwnYOLO\Data\
45 Source_Images\Training_Images\vott-csv-export\Annotations-export-new2.csv', sep=',', index=False)

```

Figure 19: Script for converting the Labelbox output into the input format

After running this python file, the next step was to convert this .csv file into the YOLO format. For this step, the first code written by Muehlemann (2019) was used. This python file is called “Convert_to_YOLO_format.py” and, as the name says, converts the file created into a file the algorithm can read. The whole code can be accessed using the following link: [https://github.com/AntonMu/TrainYourOwnYOLO/tree/master/1 Image Annotation](https://github.com/AntonMu/TrainYourOwnYOLO/tree/master/1%20Image%20Annotation). The rows where changes have to be made are from line 22 to line 30. Within these lines of code, all the names of the used input folders, the used input files, the files for the output folders and the name of the output files. As a prevention against possible errors during any calculations it is recommended to use different names for the files for each different training. The output after running this script are two text files. The first text file is stored in the “Model_Weights” folder. This file lists all the classes used for the analysis. The second one is the YOLO training file. This is again a list where each row describes one particular image. This time, the ID is the image in the corresponding source folder. After this the coordinates and the referring label (as a number) are listed. With the use of those two text files, the training of the Convolutional Neural Network can begin.

4.3 Training the CNN

Since the data needed for the training process of the Convolutional Neural Network has been converted to a format that the algorithm can work with, it is now time to start the training itself. But before the script for the training can be run, there is, according to Muehlemann (2019) one more step recommended. This is the download and the conversion of pre trained YOLO weights. These weights have been already calculated using the ImageNet 1000 data set. This database has already been trained on over 14 million images (ImageNet, n.d.) and as mentioned by Muehlemann (2019), does work very well for object identification and detection tasks. Since it is very likely that this database was not specifically trained for Japanese macaques, these pre-trained weights will not have as big an influence on the training as they would have for other classification tasks. Nevertheless, the code for downloading these pre-trained weights is a step that can ease the training process itself. The code is located in the “Training” folder and is called “Download_and_Convert_YOLO_weights.py”. For starting the training itself, the script “Train_YOLO.py” from the same folder has to be run. This code can be accessed in the corresponding folder as well as when using the link provided: https://github.com/AntonMu/TrainYourOwnYOLO/tree/master/2_Training. In the code there are several sections where changes can be made or better said have to be made. In the first lines in the code Muehlemann (2019) has written, all necessary packages and requirements to run the CNN using YOLO v3 are imported. Most of these packages, especially the ones for the YOLO v3 algorithm, are imported directly from keras. This allows the script to implement certain necessary parts from keras which are needed for a successful run of the CNN.

The next part of importance within the code are the lines 62 to 73. In these lines of code, the same as done while converting from Labelbox to YOLO format has to be done, setting the names from either the input files or the input folders. These should be changed depending on the set of training data needed for the training and the data classes for which the CNN should be trained for. After the names of all documents are set, there are only a few changes within the code left but these changes have the most influence on the success of the CNN using YOLO v3. The next parameters to change are the validation split, the amount of epochs, the batch size and the number of iterations.

Starting with the validation split, this number from zero to one defines how much of the input data set will be used for training the network and how much will be used for validating the training process. As already explained in section 3.5.2 before the implementation, the data

was split into two sets: 70% for training and 30% for testing. Now, when training the network, the 70% of data used for the training is again split into two sets of data, one for the training itself and one for validating the training process within the CNN. As per default, the value for the validation split is set to 0.1. This means that ten percent out of the training data set are randomly selected to validate the network. This means out of for example 2000 labeled images, 1800 are selected as samples for the training while 200 are selected to validate the process. According to Shah (2017) the validation set is used to fine-tune the CNN while the network never actually gains knowledge from this data. For all runs of the CNN using different amounts of input images and different classes, the value for the validation split was left at default 0.1. Changes of the validation split can be made in line 121 of the code.

The next important parameter to set before running the script of the CNN is to set the number of epochs for the training process. The number of epochs describes how often the entire training dataset is passed through the Convolutional Neural Network (Sharma, 2017b). The problem with selecting the number of epochs is that there is no guaranteed rule for choosing the right number. When selecting only a small number of epochs, the CNN will suffer from underfitting which means that there is too less learning from the data. Choosing too many epochs can result in overfitting the network, which can also cause fewer valuable results. To find the perfect amount of epochs, it is necessary to train the network with different settings in the range from ten to twenty so that both the risk of underfitting and the risk of overfitting can be minimized. To

monitor the relationship between the number of epochs and the loss value, a learning curve is often used (Brownlee, 2019). An example of a fitting learning curve showing the number of epochs with relation to the amount of loss during the calculation can be seen in Figure 20.

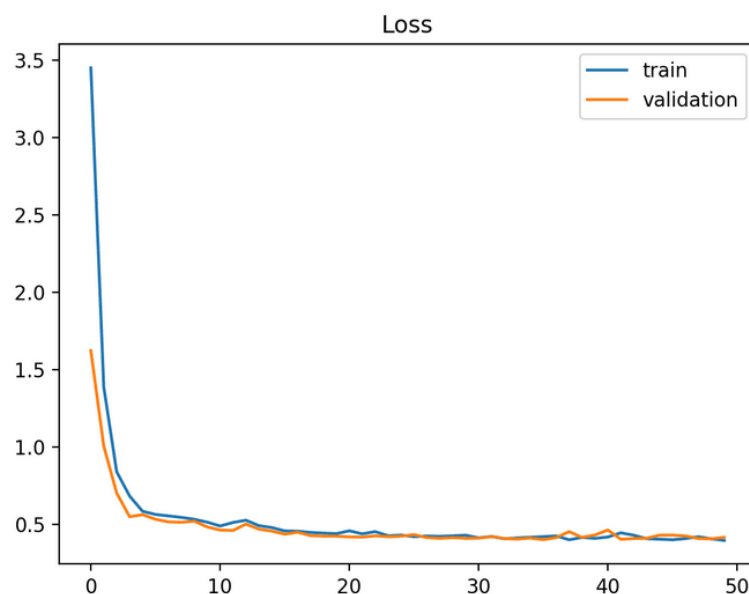


Figure 20: Example of a good fitting learning curve during the training of a CNN; adopted from Brownlee, 2019

According to GeeksForGeeks (2020) the right number of epochs is reached when the loss value of the CNN has reached a point where it starts to slightly increase again. This means that the network has reached a point, where it will not learn any new information from the data set. Therefore, within this script for the training, a so called early stopping for the CNN is necessary. This will stop the network as soon as a certain threshold for loss increase has been reached. The default value for the amount of epochs is set to 51 and can be changed at line 139 of the script. This value definitely has to be changed since 51 epochs are definitely too many epochs and risk overfitting of the network. Even with the early stopping included into the script, the risk of a slightly increasing loss value which is still lower than the threshold will still be present. For the calculations within this project, the number of epochs was set to 15. This value was chosen since higher numbers didn't show mentionable progress in the learning process of the network. Additionally, setting the number of epochs to 15 does mean that there are actually 30 epochs passed through since the value of 15 is used twice: 15 epochs for the first stage of training and 15 epochs for fine-tuning.

After setting the number of epochs for each of the two stages, it is now time to choose the right batch size for the analysis. Since an epoch tells, how often the training images are passed through the network, the number of images has now to be limited since sending all of the images through the CNN at once is too much to handle for the computer. Therefore, the data set has to be divided into several batches (Sharma, 2017b). For dividing the whole training data into several batches, a batch size has to be chosen. This value will describe how many of the images will be sent through the CNN at once. Because the training is divided into two stages, the first training stage and fine-tuning, also two different batch sizes have to be chosen. According to Brownlee (2018), there are three different ways to divide a data set into certain batch sizes:

- Batch Gradient Descent: this describes when the batch size is the same size as the training data set
- Stochastic Gradient Descent: the batch size is equal to 1
- Mini-Batch Gradient Descent: the batch size is a value between 1 and the number of images in the training data set

As mentioned before, sending all images at once through the CNN will be too much to handle for the computer, a mini-batch gradient descent was chosen for the calculations. The first batch size was chosen as 32, as it was set per default within the script of Muehleemann (2019)

in line 250. The second batch size for the fine-tuning stage has to be set in line 287 and the number 4 was chosen. This will indicate that within the fine-tuning stage, only 4 images will be passed through the network at once. Having a lower batch size refers in having more iterations within each epoch, which is needed especially for the fine-tuning stage.

The last changes that have to be made within the script are the names of the output weight files after both stages of the training. These names can be set in line 269 for the weights of the first stage and in line 306 after the stage of fine-tuning. The outputs are saved as .h5 files, which is a commonly used format for weight files of a Convolutional Neural Network. Both of those files can be used for testing the network. It can be expected that using the .h5 file after stage one of the trainings will result in worse testing of the network than after fine-tuning.

After all parameters of the code have been set, the code can be run using the Anaconda Prompt. The script by Muehlemann (2019) is written so that for each epoch, the values for the loss and the validation loss as well as the number of iterations and the runtime are displayed, which can be seen in Figure 21.

```
623/623 [=====] - 4865s 8s/step - loss: 22.1134 - val_loss: 19.8979
Epoch 17/30
623/623 [=====] - 4793s 8s/step - loss: 19.3695 - val_loss: 18.7628
Epoch 18/30
623/623 [=====] - 4799s 8s/step - loss: 18.3522 - val_loss: 17.8362
Epoch 19/30
623/623 [=====] - 4815s 8s/step - loss: 17.8058 - val_loss: 17.1674
Epoch 20/30
623/623 [=====] - 4861s 8s/step - loss: 17.3606 - val_loss: 17.2389
Epoch 21/30
623/623 [=====] - 4832s 8s/step - loss: 16.9429 - val_loss: 17.1950
Epoch 22/30
623/623 [=====] - 4841s 8s/step - loss: 16.7193 - val_loss: 16.7250
Epoch 23/30
623/623 [=====] - 4878s 8s/step - loss: 16.3728 - val_loss: 16.0276
Epoch 24/30
623/623 [=====] - 4896s 8s/step - loss: 16.2191 - val_loss: 16.4148
Epoch 25/30
623/623 [=====] - 4905s 8s/step - loss: 15.8770 - val_loss: 16.0800
Epoch 26/30
623/623 [=====] - 5078s 8s/step - loss: 15.5546 - val_loss: 15.5742
Epoch 27/30
623/623 [=====] - 4910s 8s/step - loss: 15.2907 - val_loss: 15.6519
Epoch 28/30
623/623 [=====] - 4887s 8s/step - loss: 15.2760 - val_loss: 15.6088
Epoch 29/30
623/623 [=====] - 4856s 8s/step - loss: 14.9213 - val_loss: 14.7766
Epoch 30/30
623/623 [=====] - 4862s 8s/step - loss: 14.8371 - val_loss: 15.0084
```

Figure 21: Illustration of loss and validation loss during training

4.4 Testing of the CNN

After training the Convolutional Neural Network for detecting the individual monkeys the output are two .h5 weight files which have stored the collected data for the monkey recognition. With those files the training process of the CNN can now be tested using the subsample of the data which was selected during the preprocessing steps. Those 30% of the original data set will now be used to try to identify the monkeys within the images the network has never seen before. For testing the network, the script “Detector.py” has to be run. This script is used to validate if the training is valuable. The code is accessible via the link: [https://github.com/AntonMu/TrainYourOwnYOLO/tree/master/3 Inference](https://github.com/AntonMu/TrainYourOwnYOLO/tree/master/3%20Inference). The input files and folders as well as the output files used for the script have to be set in the lines 36 to 50. As input the weights of the trained YOLO model as well as the classes of the monkeys are used. Another important setting for the use of the script can be found in line 124. In this section, the minimal confidence value can be set. This value is showing, how confident the trained algorithm is while detecting each of the classes. It ranges from 0 to 1 where 1 means that the network is completely sure that this class is detected correctly. A high confidence value is indicating that the algorithm is sure that it detected the correct label. Lower confidence values have more of a mixed meaning since they can be correct, but the algorithm is not as sure as it is when addressing higher confidence values. If setting a higher confidence value threshold, less result bounding boxes are detected but it can be assumed that the number of correct detections can be increased. While running the script each image of the test data set images is passed through the detector and the weight file created with YOLO v3 is detecting the corresponding classes in each of the images. An example of how this looks like using Anaconda Prompt can be seen in Figure 22.

```
C:\Users\Benutzer1\Desktop\2nd_try\TrainYourOwnYOLO\Data\Source_Images\Test_Images\Sandra02311.jpg
(416, 416, 3)
Found 2 boxes for img
Sandra 0.99 (1237, 440) (2667, 2146)
Sandra_face 1.00 (1966, 828) (2304, 1360)
Time spent: 0.481sec
```

Figure 22: Illustration of the detection of an image while testing

The results using the “Detector.py” script are all of the input images with the detected bounding boxes for each of the classes. Those bounding boxes are different in color and show the name of the label and the value of confidence of the detection. Additionally, those attributes and the coordinates of the bounding boxes are stored in a .csv file. These files can be used for evaluating the individual results for all the labels and for comparing the results of the different trainings. The images with the bounding boxes on them are visually showing the results and can be easily used to understand what the algorithm has detected. Such images are the typical outcomes of a trained CNN. One image of the resulting detection can be seen in Figure 23.

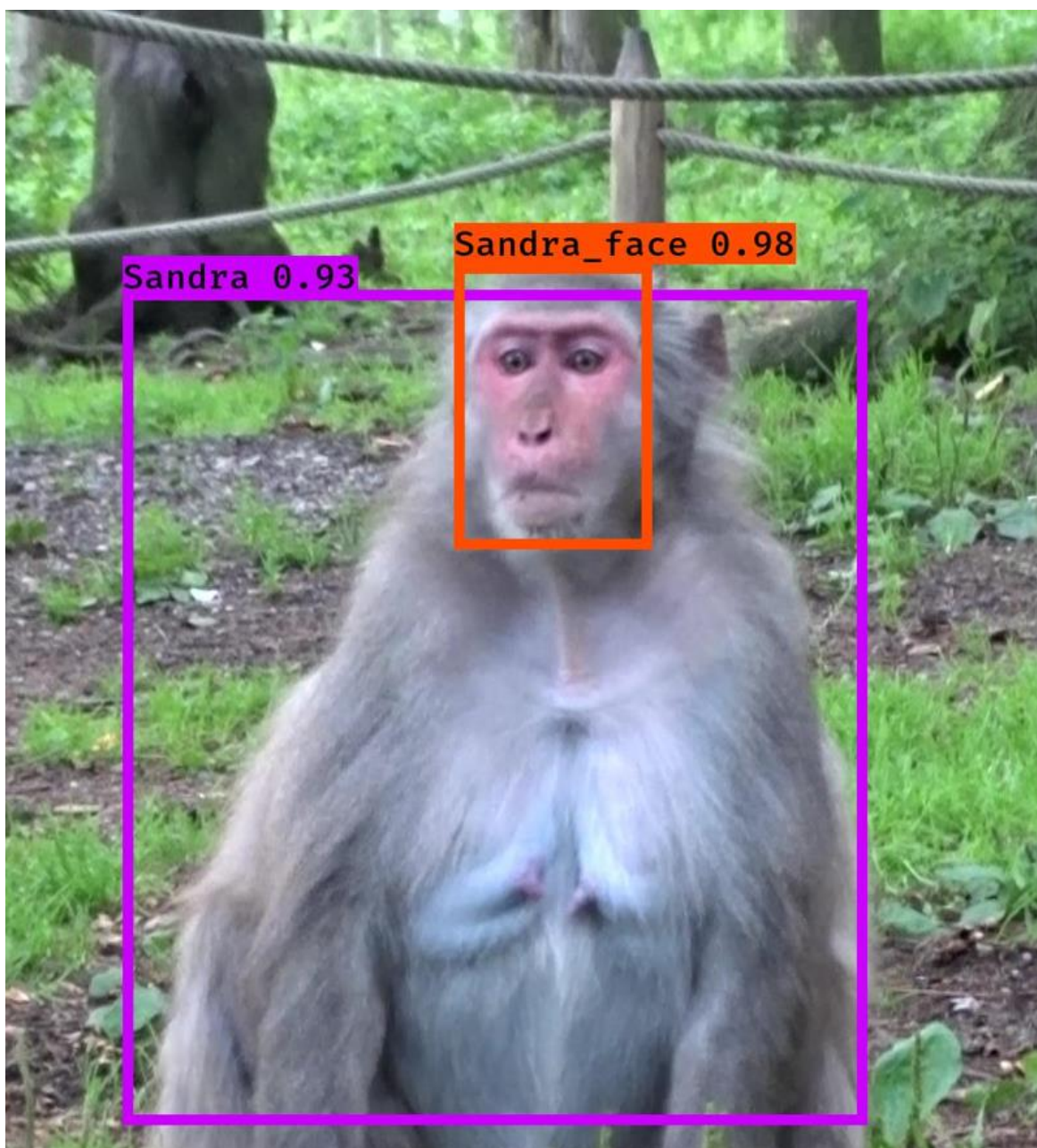


Figure 23: Example of a successful identification

5. Results

Within this section of the thesis the results of the trained networks using YOLO v3 are shown. This section is divided into several subsection. The first one will briefly show the results of the proof of concept. Out of these first results conclusions for the further training runs have been made. The results of these training runs will be shown in the second section. Those results will then be evaluated using the method described in section 3.7. After the evaluation of the results the influence of the value of confidence will be shown. This will demonstrate how important the confidence value calculated by the CNN is. The last part of this section of the thesis will describe the results using the acquired data within this project. This will display the possibilities of the work with Convolutional Neural Networks and will show what the CNN is capable of.

5.1 Proof of concept results

In this smaller section of the results, the first look will focus on the training of the proof-of-concept data sets. As previously explained, in the first trainings only three out of ten monkeys were used for the identification. Those three monkeys were selected due to the differences within the monkeys and the amount of data that was collected. There was also the attempt to include an overall monkey class. The classes were split into the three main body parts of the monkey: face, head and body. The trainings that took place were one with all the classes and all three monkeys, a training for each of the body parts individually and a final training only using images from behind. The results will be shown in the next paragraph.

The first look will be on the output images from those trainings. Figure 24 is displaying one of the best images from each monkey.

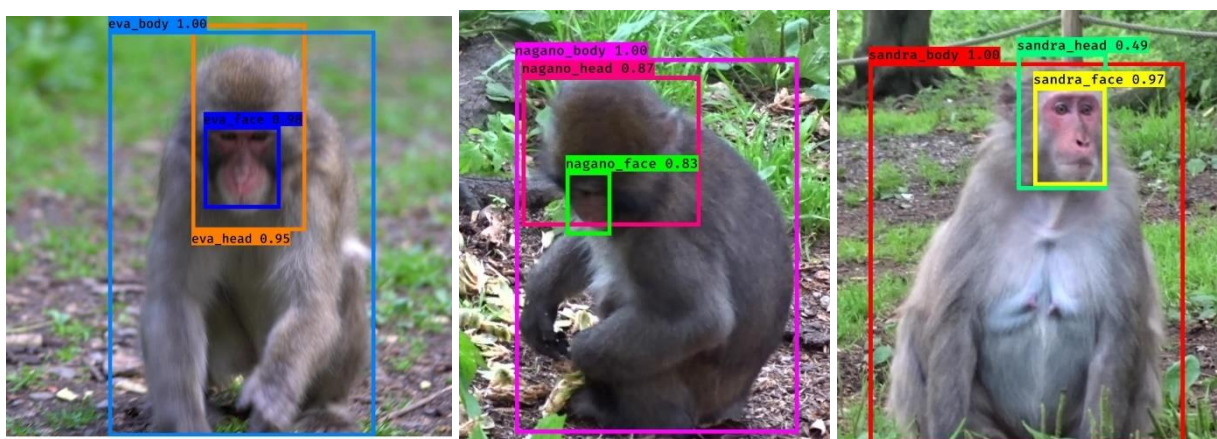


Figure 24: Successful identification of the proof of concept (from the left: Eva, Nagano, Sandra)

Those examples of the three monkeys show how an almost perfect identification of the monkeys and a prediction of the bounding boxes should work. For those three classes for each monkey, the identification works pretty well. Since those results are only for the proof of the concept, it is not necessary to evaluate them furthermore. Overall it can be said that those predicted bounding boxes are detecting the different body parts.

For the next comparison of those three monkeys, the Convolutional Neural Network was trained three times where each of the trainings focused on detecting one particular body part.

As a visual result, the following Figure 25 shows the results of the detection of each of the body parts for the same image as well as the detection of all three labels trained together.

As it can be seen in Figure 25, all of the different trainings result in a good detection of the different parts of the body. This is of course only one example out around 620 images which were used for testing the algorithm. Nevertheless, it shows that the described way of identifying the monkeys works and delivers reasonable results and specifically identifies all three monkeys. The point which still has to be mentioned is that in most training pictures classes like the head and the body also include the face of the monkey. This means that without training the network

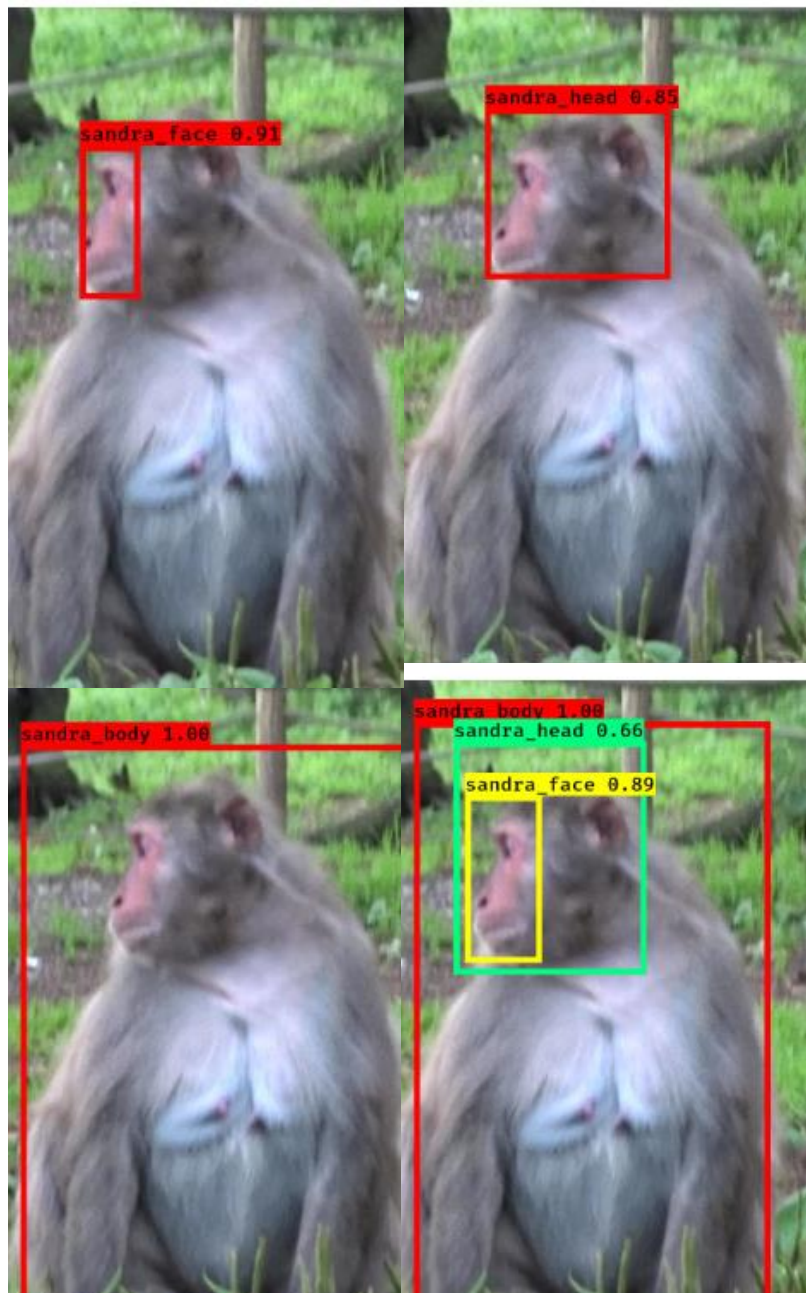


Figure 25: Example image: comparison of the different training data sets during proof of concept (from top left to bottom right: only face, only head, only body, all three)

only on images from behind the monkey, it can never be said that the network can detect the body of the monkeys because it could be only looking at the face as well. When looking at all the labels created by the different data sets, it can be recognized that overall the identification of the face of the monkey worked the best. This was also expected since this body part is the most significant part of the monkey and most expert also use the face to identify the monkey based on special parts of the face. The detection of the head and the body was faultier than the detection of the face. This is surprising since the head is basically just a bigger bounding box around the face, but the algorithm had already more problems detecting the head in the right shape. An interest conclusion out of those trainings is that an overall monkey class is most of the times more disturbing than helpful when identifying the individuals. The classes mostly detected the known monkeys which is basically not wrong but was disturbing the other classifications and also misleading. Furthermore, the confidence of most predictions was really high while the confidence values for the overall monkey class was pretty low. This proves the initial assumption that the identification will most likely succeed if the monkey is alone in the image. Therefore, these classes will not be used for the further analysis.

To cover the difficulty with knowing if the network will also detect the monkeys if the face cannot be seen in the image the last training for proving the initial ideas was a training using images from behind. When the videos were recorded, the main focus was on the face. Therefore, there is not as many training data available were the monkeys can only be seen from behind. Results of this training can be seen in Figure 26.

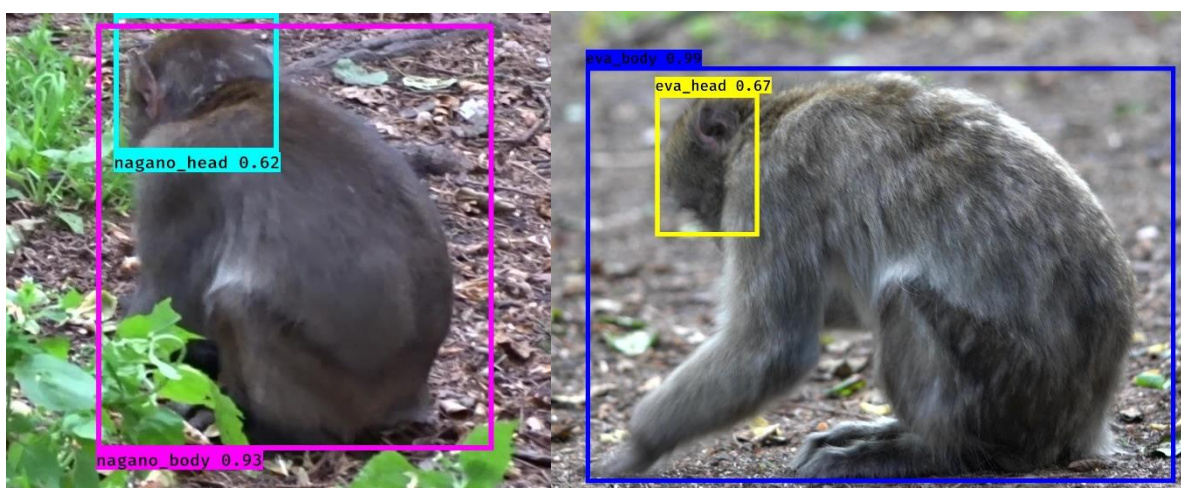


Figure 26: Result of the training from behind

The images show almost the same good prediction as the images not specialized on the monkeys from behind. Nevertheless, it is good to see that also when not trained on detecting the face, the CNN is capable of correctly identifying the monkeys.

5.2 Results using all ten monkeys

For training a Convolutional Neural Network which is capable of detecting all ten monkeys, the conclusions made during the proof of concept were adapted to succeed using all ten monkeys. In most cases using YOLO v3 adding more objects to detect and therefore more labels will decrease the precision of the network and therefore the overall performance. Since this is also the case when increasing the number of monkeys from three to ten, it can be expected that the network will not be able to detect the monkeys as good as the previous training runs. Furthermore, it can be expected that because of different amounts of training and testing data for the individual monkeys the prediction of the bounding boxes and the identification of the monkeys with less data will not work as good as for the monkeys with more data available. The section for the results is divided into three subsections. The first one will show the results of the calculations using 15 epochs, the second one will show the results using 25 epochs and the third part will explain the results of using balanced data sets for training and testing.

5.2.1 Results using 15 epochs

As it was already mentioned in the previous section, different data sets and different labels have been used to detect all ten monkeys. The first two detections were done using all images from the training data set. These calculations were done using 15 epochs. As a first impression of the results there will be some sample images for all ten monkeys comparing the face recognition and the body recognition. The overall comparison of the different trained runs will

be shown in the next section when the results will be evaluated. The first visual comparison is shown in Figure 27.

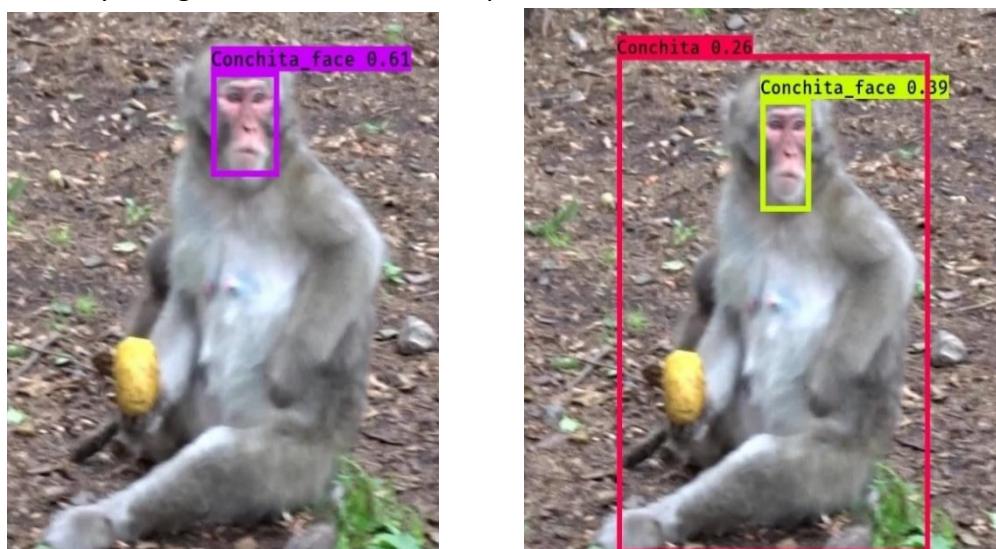


Figure 27: Result using 15 epochs: difference of the confidence (left: only face recognition; right: body and face recognition)

This picture of Conchita shows the biggest difference when using only the labels for the face compared to using both face and body labels. Although in both images the monkey was correctly identified, the confidence in the image where both labels can be seen is relatively low compared to the confidence of 0.61 in the detection using only the face. This difference has been detected in a lot of cases during the analysis of the images. On the opposite, there are also a lot of images where the detection of the face and the body have the same or even a higher confidence as the prediction box of the solution only trained on the faces. An example of this can be seen in Figure 28.

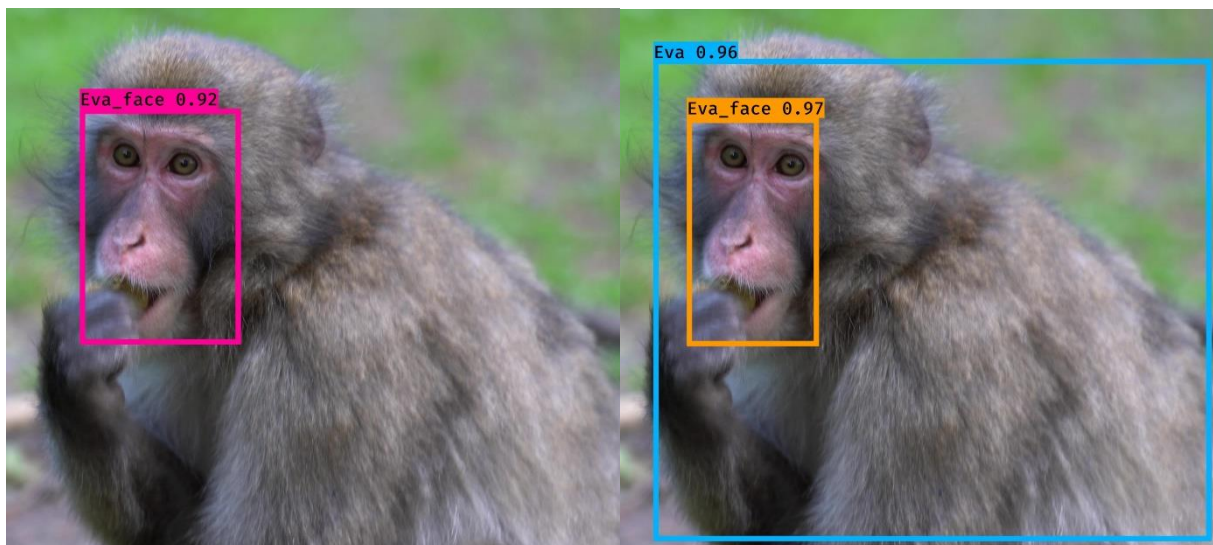


Figure 28: Result using 15 epochs: perfect detection (left: only face recognition; right: body and face recognition)

As it can be seen those results are close to perfect. Such an identification of the monkey is what the trained network using YOLO v3 is capable of. This is the case in a lot of images which leads to a high precision of the CNN.

Since in most of the images the monkeys are pretty close to the camera and can be seen alone, it is also interesting to see if the monkeys will be detected in other images as well. The first case shows how an image with no monkey in it is detected. This is shown in Figure 29.

As shown, there is no monkey detected in the image. This is really good since the network is only trained on monkeys but is looking for them and is not detecting a monkey in every single image. If a monkey would be detected within such an image, it will decrease the

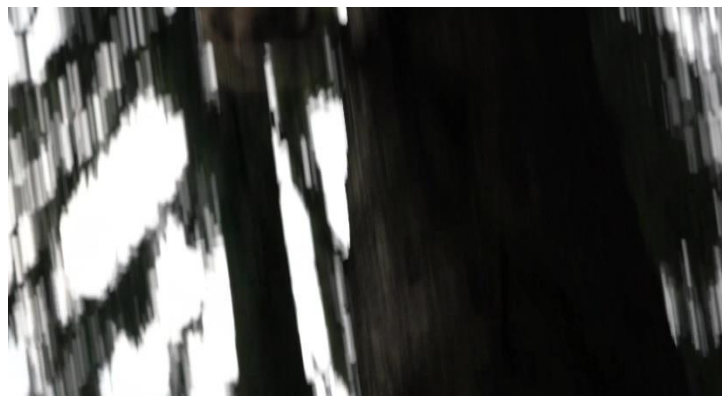


Figure 29: Example: Correct handling of an image without monkeys

precision of the network because it detected one but there is no monkey in the image (False Positive). Such cases are very limited but sometimes the network detects a monkey in such images. Since those results will influence the evaluation, they will be taken into account later during the evaluation. Another important question is how the network detects monkeys that are not one of the ten monkeys the network was trained for. Since the overall monkey class was cut off during the proof of concept, it can be expected that the monkeys will still be detected using one of the existing labels. An example of such a recognition of an unknown monkey can be seen in Figure 30.



Figure 30: Recognition of unknown monkeys within the images

In this image of Manfred it can be seen that there are three monkeys within the image. The monkey identified as Manfred is indeed Manfred, but the other two monkeys are not known by the network. This can lead to one of two cases which can both be seen in this image: the unknown monkey is either detected as another monkey the network is trained for or is not detected at all. This leads to a lot of mislabeling within such images which will also influence the evaluation.

Nevertheless, such images are more an exception than the usual when looking at the whole testing data set. Since displaying all of the 1464 images would unnecessary, the precision of each class, which is the number of correct labels divided by the number of total labels detected for each class, as well as an average value of the precisions from all classes will be shown in

Table 4. Displaying the accuracy of the network would be misleading for understanding the results since the accuracy describes the number of correct labels divided by the number of total labels. This is manipulating the data because of the different amounts for each class which will benefit majority classes and discriminate the minor classes with less data. This table will only be an overall view on the results since the detailed evaluation will be shown within the next chapter.

Table 4: Results using 15 epochs for the training

	Precision: face and body labeled	Precision only face labeled
Conchita	0.886178862	--
Conchita_face	0.812121212	0.574545
Eva	0.522058824	--
Eva_face	0.582524272	0.471698
Jessy	0.341463415	--
Jessy_face	0.222222222	0.125
Kate	0.373417722	--
Kate_face	0.346153846	0.314286
Louis	0.41991342	--
Louis_face	0.572649573	0.307692
Manfred	0.647398844	--
Manfred_face	0.676470588	0.726316
Maya	0.335766423	--
Maya_face	0.518072289	0.476744
Nagano	0.459876543	--
Nagano_face	0.61038961	0.385135
Sandra	0.334246575	--
Sandra_face	0.724489796	0.740741
Zarah	0.439393939	--
Zarah_face	0.936170213	0.717949
Average	0.538048909	0.484011

As it can be seen in Table 4, the precision for the labels detecting the face is higher than for the detection of the body. What is surprising is that for most of the classes the precision is higher when face and body were labeled than when the network was only trained using the faces of the monkeys. Looking at the precision for each class it can be assumed that this value shows how often a label is detected correct when it is detected, e.g. if the precision has a value of 0.6 it means that 60% of the time when a label is detected, it is correct. Nevertheless, the precision is not the only important factor describing the results of a CNN using YOLO v3 since

it only shows the percentage of correctly detected labels in relationship to the total amount of detected labels, not to the amount of labels it should have detected. Therefore, further analysis is necessary to calculate the with respect to the total number of ground truth bounding boxes within the test data set. This will be evaluated within section 5.3 when evaluating these results.

5.2.2 Results using 25 epochs

The next results are showing the calculations using 25 epochs instead of 15 epochs during the training process. This decreases the loss and the validation loss of the CNN and should change the precision of the network. The results are expected to be a bit better than the ones for the calculations with 15 epochs, but it will be important to see if the amount of time these calculations took will result in better solutions.

The first big difference that can be seen when looking at the different results is the enormous difference in labels with a high confidence value. While using 15 epochs for the calculations resulted in around 660 labels out of 2800 with a confidence higher than 0.9, the results using 25 epochs had over 1100 labels with a confidence over 0.9. This is an important difference because correct labels with a high confidence are the best way to identify the monkey. An example of such a better confidence value can be seen when detecting Conchita. The differences are shown in Figure 31.

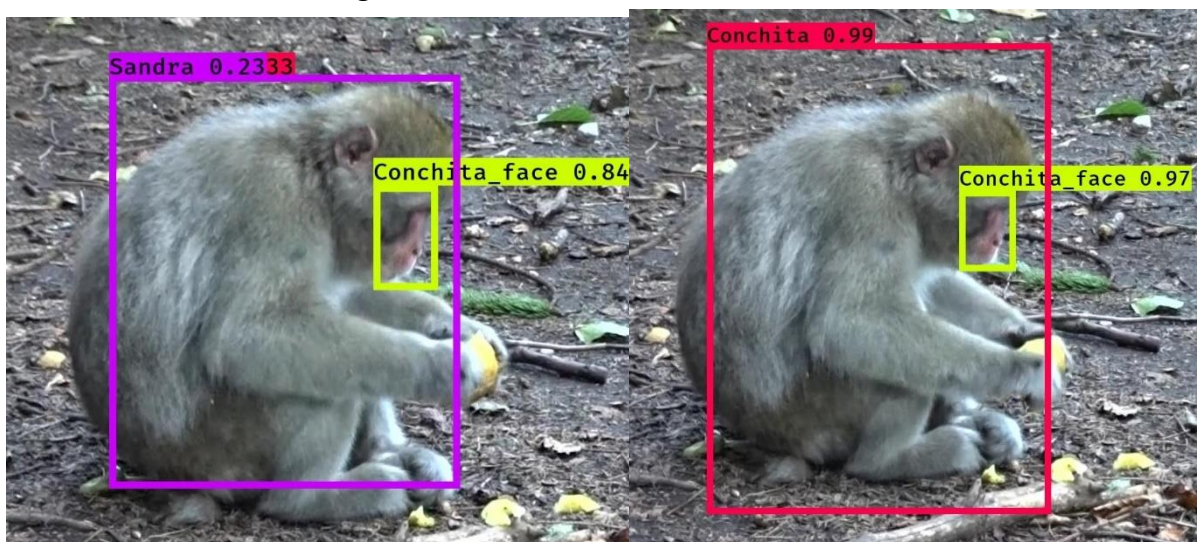


Figure 31: Difference using 15 and 25 epochs for training (left: 15 epochs, right 25 epochs)

This is a good example of a way better detection of Conchita which can also be seen when looking at the whole data set, especially for Conchita. The negative part of using more epochs is that it can happen that the model is overfitted. This can also happen for single classes and

not the whole dataset. And this has happened for three monkeys of this data set. Nagano, Jessy and Zarah have really bad results. Those labels were not detected very often with Zarah's body not even detected once despite having enough training data. This has not happened while using 15 epochs where the data set is very well distributed through all the labels.

Again, looking at all images individually would take too much time. Therefore, the output .csv file generated by the algorithm was used to create another table which shows the precision of each of the classes using 25 epochs for the training process and the average of all the classes. These precisions are shown in Table 5.

Table 5: Results using 25 epochs for training

	Precision: face and body labeled	Precision: only face labeled
Conchita	0.636094675	--
Conchita_face	0.757847534	0.728155
Eva	0.44973545	--
Eva_face	0.57	0.446043
Jessy	0.333333333	--
Jessy_face	0.285714286	0.189189
Kate	0.348101266	--
Kate_face	0.53125	0.5
Louis	0.38	--
Louis_face	0.374331551	0.40884
Manfred	0.414473684	--
Manfred_face	0.538461538	0.528571
Maya	0.602739726	--
Maya_face	0.661764706	0.61194
Nagano	0.895833333	--
Nagano_face	0.790697674	0.45
Sandra	0.386462882	--
Sandra_face	0.67037037	0.788462
Zarah	0	--
Zarah_face	0.857142857	0.870968
Average	0.524217743	0.552217

The values seen in this table show that the precision for certain classes is very high which indicates that almost every label predicted by the algorithm was correct. The difference to the precisions using 15 epochs is that the training specifically for the face of the monkeys delivered better precisions than the one also trained for detecting the body. Difficulties arose when

trying to identify Zarah. As it can be seen the body was not detected once while despite having a high precision also the face of Zarah was not detected very often. Therefore, the precision value is high but the result for the further evaluation is expected to be very low since only in a few images the monkey was detected. Nevertheless, the precision of some of the classes is quite high which is indicating that if such a label is detected, chances are high the detection is right.

The other monkey with problems in every detection is Jessy. Jessy is the monkey with the least amount of training images. This means that the algorithm is more likely to detect other monkeys instead of Jessy because it is better trained for detecting the other monkeys and not Jessy. To balance this, the next section will show the results of using an equally distributed amount of training images for all the monkeys. This will display if the problems with Jessy are because of the lack of data or if there are other difficulties with identifying Jessy.

5.2.3 Results using balanced data

The last trainings that took place were runs using the equal amount of training data for each of the monkeys. This will minimize the challenges of the prior trainings since using the same amount of images for each monkey, no monkey is prioritized during the learning process. Since the amount of data for Jessy is the lowest, the number of images extracted from the video will be the limit for the selection of the images from the other nine monkeys. The images for the other monkeys were then randomly selected from the pool of images that are available. Another category for the selection of the images was that both labels, face and body of the monkey, can be seen within the image. The number of images selected for training is 58. This results in the use of 25 images for testing since the split of 70% for training and 30% for testing should still be intact.

A problem arising when dealing with a smaller amount of data is that the model can be easily underfitted. Since the selection of the validation set from the training data set is randomly, 10 percent of the data are used for validation. This could mean that from one monkey more images are used for validation which could lead to less precise detection of certain classes. The biggest difference to the predictions from prior calculations is the confidence of the prediction bounding boxes. While for some monkeys the identification worked almost perfectly (Figure 32), other monkeys have had more problems to be detected.

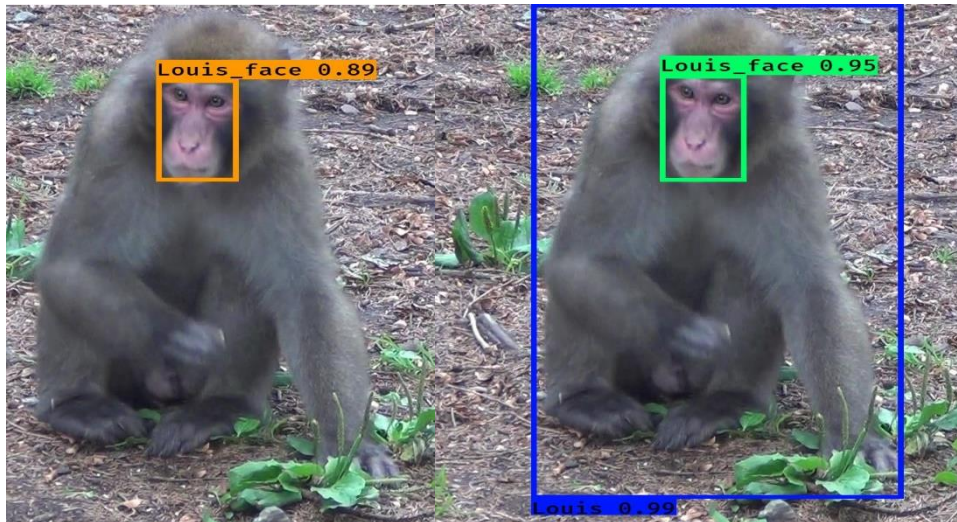


Figure 32: Good result using balanced data for training

The identification for Louis was very good but the monkey was also detected in a lot of other images despite not being in the image. Other monkeys like Zarah had again problems to be identified with only a few labels detected overall. This can also be seen in Table 6 where the precision for each of the classes is displayed.

Table 6: Results using balanced data for the training

	Precision: 58 images with face and body labels	Precision: 58 images with only face labels
Conchita	0.425532	--
Conchita_face	0.448276	0.375
Eva	0.705882	--
Eva_face	0.65	0.5625
Jessy	0.777778	--
Jessy_face	0.5	0.458333
Kate	0.488889	--
Kate_face	0.5	0.785714
Louis	0.271739	--
Louis_face	0.370968	0.359375
Manfred	0.190476	--
Manfred_face	0.432432	0.703704
Maya	0.609756	--
Maya_face	0.5	0.543478
Nagano	0.944444	--
Nagano_face	0.6	0.25
Sandra	0.8	--
Sandra_face	1	0.708333
Zarah	0.666667	--
Zarah_face	0	1
Average	0.544142	0.574644

The results shown in Table 6 differ a lot from the previously shown comparisons. The differences between the individual monkeys are relatively big since for some monkeys the detection worked better than before while for other monkeys, the results are not satisfying. Again, only looking at the precision of each of the classes does not tell the whole story. For Sandra and Zarah the precision is a really high value, but the number of detected labels compared to the number of total images is quite low e.g. Sandra was detected 7 times, 7 of those labels are correctly detected but there are 25 images of Sandra in the testing data set. This means that although Sandra has a precision of 1, only within 7 out of 25 images the monkey was detected. This can also be stated as follows: Sandra is detected about 30% of the time but if she is detected, the detection is right.

Other results only looking at the precision are also misleading since for Louis and Maya the precision values are below average respectively a bit above average, but those monkeys are also detected a lot of times. As an example, Maya was detected in every of the 25 images where Maya can be seen, but the monkey was also detected within 16 other images.

Such examples are the reason why the results have to be evaluated. This evaluation will be described within the next chapter of the thesis.

5.3 Evaluation of the results

As in the previous section described the results showing the precision of each trained network can be seen as a first impression of the results. But it was also shown that only using the precision to represent the performance of the network is not enough since sometimes good precision does not indicate a good performance. Therefore, the already in section 3.7 described method for evaluating the results will be used to show the capability of the trained networks to identify the monkeys.

The evaluation will contain the previously calculated precision, the recall (also called sensitivity) and the area of the intersection of the predicted box with the ground truthing box over the union of the two boxes (IoU). Additionally, an according to the confidence value of each detection ranked table for calculating the precision and recall after every image and to draw a precision x recall curve and the calculation of an interpolated Average Precision (AP), which is the area under the previously drawn curve will be part of the evaluation. The last step will be to calculate the mean Average Precision (mAP) using the AP of all twenty classes.

The final mAP will be calculated for all of the six previously mentioned training runs using data for all ten monkeys. Since displaying the calculations for all steps for each of the six trained networks will not be beneficial for the readability of the thesis the whole evaluation process will be shown using the calculation of face and body with 15 epochs. The final Average Precision as well as the mean Average Precision will be displayed for all training runs. These results can then be compared to the mAP of YOLO v3 shown by Redmon and Farhadi (2018) which have tested the algorithm on the COCO dataset.

For evaluating the results the first part is to compare the predicted bounding boxes detected by the algorithm with the from experts labeled ground truthing bounding boxes. For this step, the output from the algorithm was compared with the list of the ground truthing boxes to bring the corresponding bounding boxes for each detected label together so the Intersection Over Union can be calculated. To fasten this process a script was written which compares the predicted bounding boxes with the list of ground truth bounding boxes and if the classes are matching is additionally calculating the Intersection over Union. The script written in python can be seen in Appendix B.

Every label which is inserted into the script can have one of three outcomes: either the bounding box is located in both the prediction and the corresponding ground truth image, or it is not within the image where it would be expected. The third possibility is that a label was

detected within an image where even the expert did not identify the monkey. As a visualized way to understand the different cases, the following Figure 33 was created.

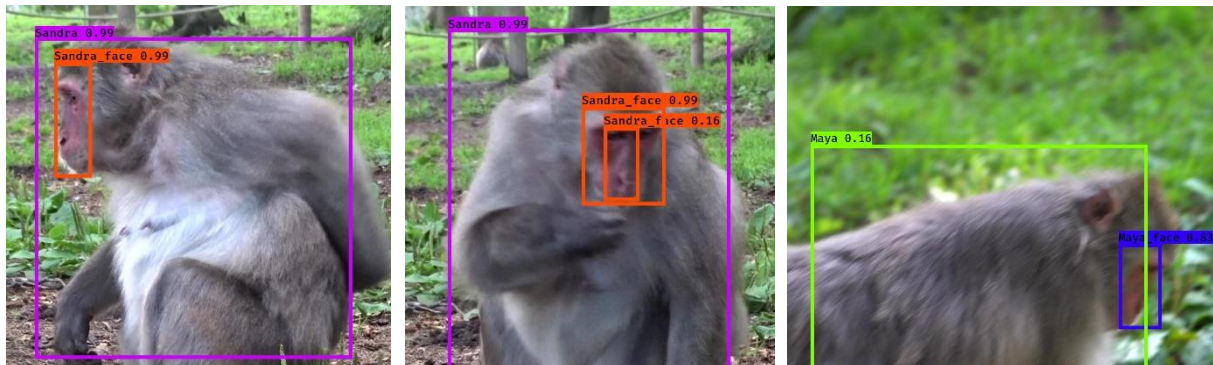


Figure 33: Illustration: cases of True Positive (left), False Positive (middle) and False Negative (right)

In the image on the left side both the face and the body of Sandra were detected correctly, and the IoU value is higher than the threshold. Therefore, both labels count as True Positive (TP). In the middle image, the bounding box for Sandra's body is correctly detected, henceforth TP. The face is detected twice where the IoU value from one bounding box is over the threshold and from the other it is below. Those are classified as one TP and one False Positive (FP). In the left image of Sandra, Sandra is not detected at all. This will count as a False Negative for both face and body. Since instead Maya is detected in the image but Maya is not there, those labels count as FP. Those three possibilities have to be checked for each of the predicted labels. A portion of the output can be seen in Figure 34.

image	label	label_number	confidence	IoU
Louis_200781.jpg	Louis	12	0.995147705	0.752562
Louis_202071.jpg	Louis	12	0.994451344	0.888814
Louis_202461.jpg	Louis	12	0.993652344	0.66159
Louis_202131.jpg	Louis	12	0.993493855	0.738455
Louis_202521.jpg	Louis	12	0.99305141	0.805226
Manfred_401201.jpg	Louis	12	0.991912186	-1
Louis_202251.jpg	Louis	12	0.990920901	0.802682
Louis_201921.jpg	Louis	12	0.990237713	0.914491

Figure 34: Addition of the value for Intersection Over Union to the result table

The output after running the script is showing the predicted labels with the corresponding class as well as the, if it was possible, calculated value for the IOU. After that each class of the network has to be handled separately. Therefore, the whole output was divided into the data for each of the classes. The split-up data for each class is then ordered according to the value

of confidence for each label. After that, the values for precision and recall can be calculated while iterating through the labels. This will indicate that labels with high confidence values are processed first while labels with low confidence values are processed last. For the evaluation the order of the labels is directly influencing the final result. In most cases the label with the highest confidence value is a correct detection. This will mean that the precision after this label is 1 since one out of one label was detected correctly. The value for recall is relatively low since it is 1 divided by the total number of ground-truthing boxes. After looking at each label one after another those values change with the precision getting lower if a label was detected wrong (FP) while the recall is increasing depending on how many labels were detected correctly. These value changes can be seen in Figure 35 where a portion of the calculation table is displayed.

image	label	label_nun	confidence	IoU	TP	FP	Acc TP	Acc FP	Recall	Precision
Louis_201951.jpg	Louis	12	0.990035	0.731515	1	0	1	0	0.010204	1
Louis_201921.jpg	Louis	12	0.989767	0.64995	1	0	2	0	0.020408	1
Louis_202071.jpg	Louis	12	0.988683	0.674175	1	0	3	0	0.030612	1
Louis_202341.jpg	Louis	12	0.986816	0.737639	1	0	4	0	0.040816	1
Louis_202521.jpg	Louis	12	0.986484	0.65914	1	0	5	0	0.05102	1
Louis_202461.jpg	Louis	12	0.986351	0.603672	1	0	6	0	0.061224	1
Louis_201561.jpg	Louis	12	0.984486	0.711161	1	0	7	0	0.071429	1
Louis_201981.jpg	Louis	12	0.983984	0.774951	1	0	8	0	0.081633	1
Louis_201591.jpg	Louis	12	0.983704	0.695239	1	0	9	0	0.091837	1
Louis_202011.jpg	Louis	12	0.983516	0.780206	1	0	10	0	0.102041	1
Louis_202251.jpg	Louis	12	0.983051	0.750903	1	0	11	0	0.112245	1
Louis_202041.jpg	Louis	12	0.982683	0.718892	1	0	12	0	0.122449	1
Louis_200361.jpg	Louis	12	0.980255	0.843685	1	0	13	0	0.132653	1
Louis_202101.jpg	Louis	12	0.980093	0.704925	1	0	14	0	0.142857	1
Louis_201741.jpg	Louis	12	0.97939	0.789045	1	0	15	0	0.153061	1
Louis_201891.jpg	Louis	12	0.979162	0.680108	1	0	16	0	0.163265	1
Louis_202491.jpg	Louis	12	0.978757	0.768699	1	0	17	0	0.173469	1
Louis_201201.jpg	Louis	12	0.978195	0.740246	1	0	18	0	0.183673	1
Louis_202161.jpg	Louis	12	0.977448	0.840334	1	0	19	0	0.193878	1
Louis_202131.jpg	Louis	12	0.976059	0.776949	1	0	20	0	0.204082	1
Louis_201441.jpg	Louis	12	0.973577	0.747035	1	0	21	0	0.214286	1
Louis_202191.jpg	Louis	12	0.973317	0.790361	1	0	22	0	0.22449	1
Louis_202281.jpg	Louis	12	0.969442	0.768006	1	0	23	0	0.234694	1
Louis_202371.jpg	Louis	12	0.968973	0.670127	1	0	24	0	0.244898	1
Louis_201831.jpg	Louis	12	0.966968	0.766025	1	0	25	0	0.255102	1
Louis_201111.jpg	Louis	12	0.965652	0.777016	1	0	26	0	0.265306	1
Louis_201681.jpg	Louis	12	0.965499	0.70955	1	0	27	0	0.27551	1
Louis_201531.jpg	Louis	12	0.964911	0.632306	1	0	28	0	0.285714	1
Louis_201651.jpg	Louis	12	0.964096	0.691304	1	0	29	0	0.295918	1
Louis_202641.jpg	Louis	12	0.96205	0.797408	1	0	30	0	0.306122	1
Manfred_400211.jpg	Louis	12	0.961183	-1	0	1	30	1	0.306122	0.967742
Louis_201261.jpg	Louis	12	0.960996	0.725256	1	0	31	1	0.316327	0.96875
Louis_202551.jpg	Louis	12	0.96077	0.709777	1	0	32	1	0.326531	0.969697
Louis_200301.jpg	Louis	12	0.959839	0.818425	1	0	33	1	0.336735	0.970588
Louis_201621.jpg	Louis	12	0.958476	0.595629	1	0	34	1	0.346939	0.971429

Figure 35: Best results of the detection of Louis

As shown in Figure 35 the value for recall and precision do change while going through the labels. The section shown from Louis' calculation are the 35 labels with the highest value of confidence. The columns named "Acc TP" and "Acc FP" are counting the number of True

Positives and False Positives. The end of the list will show the final value for both precision and recall. The last few rows of the calculation can be seen in Figure 36.

image	label	label_nun	confidenc	IoU	TP	FP	Acc TP	Acc FP	Recall	Precision
Manfred_403271.jpg	Louis	12	0.15529	-1	0	1	97	128	0.989796	0.431111
Manfred_400931.jpg	Louis	12	0.153844	-1	0	1	97	129	0.989796	0.429204
eva01981.jpg	Louis	12	0.15253	-1	0	1	97	130	0.989796	0.427313
Conchita06661.jpg	Louis	12	0.152327	-1	0	1	97	131	0.989796	0.425439
Conchita00331.jpg	Louis	12	0.152308	-1	0	1	97	132	0.989796	0.423581
Manfred_400421.jpg	Louis	12	0.152289	-1	0	1	97	133	0.989796	0.421739
Manfred_403031.jpg	Louis	12	0.151568	-1	0	1	97	134	0.989796	0.419913

Figure 36: Last few rows of Louis' result table

As it can be seen 97 labels have been detected correctly while 134 labels were detected wrong. This is why the precision is under 0.50. This value can be influenced using a threshold for the value of confidence. As an example choosing the threshold of 0.5 for the value of confidence, only 52 wrong labels would be taken into account while 88 correct labels would still be over the threshold. This proves how the confidence value can influence especially the value for the precision. The important part is to be consistent about the thresholds over the whole data set and not to choose thresholds so that the individual classes would only benefit from the selection.

The two columns representing the values for precision and recall are used to draw a precision x recall curve. This curve demonstrates how those two values change during the process. The x axis shows the value for the recall and ranges from 0 to 1 while the y axis is displaying the precision, also ranging from 0 to 1. This has to be done for every class individually since the curve is needed to calculate the area under the curve which is the value for the Average Precision (AP). The precision x recall curve for the above shown data of Louis' body trained using 15 epochs can be seen in Figure 37.

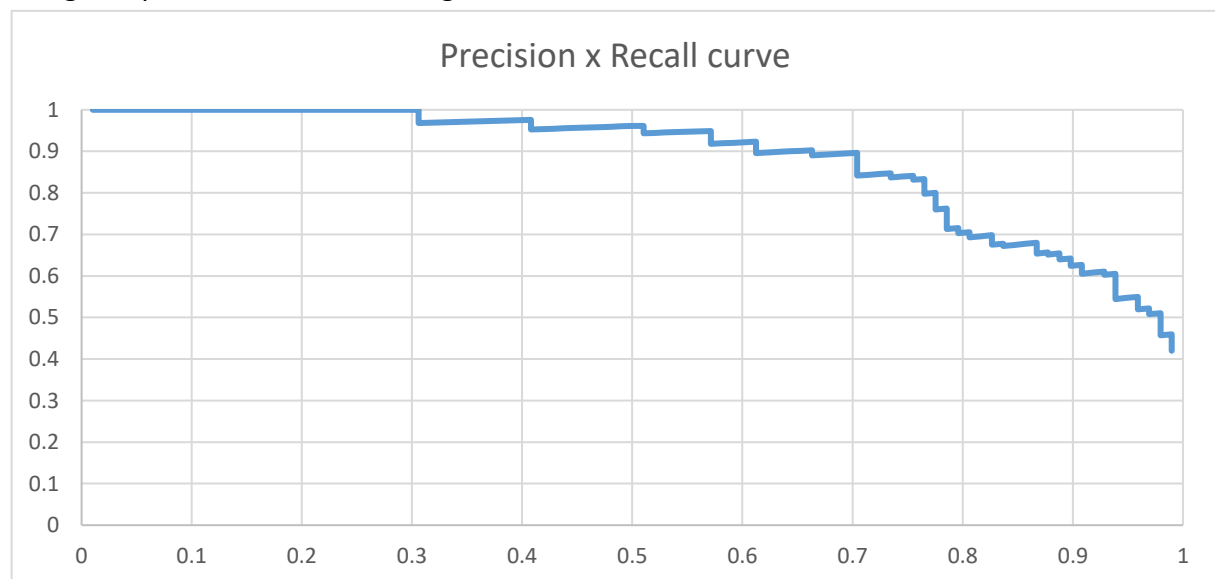


Figure 37: Precision x Recall curve using the ordered List of labels

This curve shows the progress of the precision and recall values while iterating through the labels. Additionally, this curve can now be used to calculate the AP by calculating the area under the curve. Since this can be quite complicated when dealing with a lot of labels, the complete data can be adapted to use the so called 11-point interpolation approach. This technique is using 11 interpolated points for the recall values starting at 0 and increasing by 0.1. This will lead to eleven points within the graph where the corresponding precision values can be used to calculate an interpolated area under the curve. The graph including the 11 points for the calculation can be seen in Figure 38.

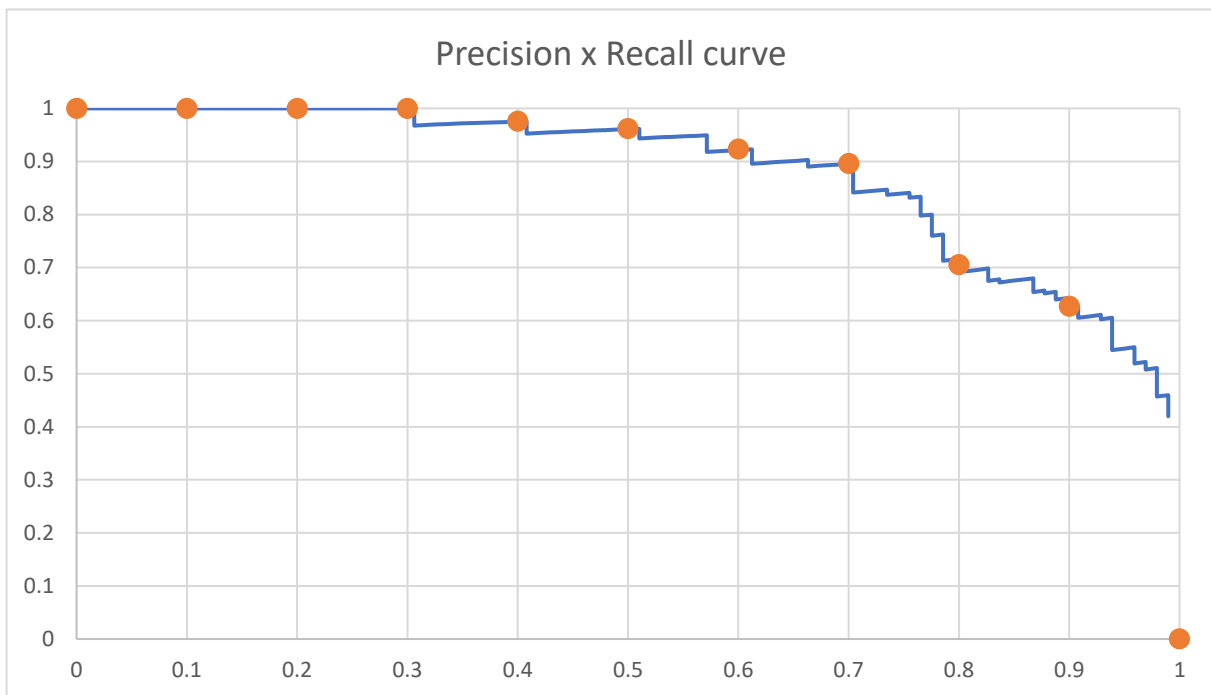


Figure 38: Precision x Recall curve with the points for the 11-point interpolated evaluation

To get the corresponding points for each of the recall values, the maximum value going from every individual point to the right is selected. In this case, the interpolation does not benefit the calculation for the area under the curve. For the other classes the interpolation is more beneficial which is again balancing the results when looking at the overall results. With those eleven points the final value for the Average Precision for this class can be calculated. This is done using the formula for the 11-point interpolation approach. The value for the AP is calculated as follows:

$$AP_{11} = \frac{1}{11} * (1 + 1 + 1 + 1 + 0.97 + 0.96 + 0.92 + 0.89 + 0.70 + 0.62) = 0.82$$

For the calculation of the mean Average Precision (mAP), the values for the AP have to be calculated for each individual class for each results set introduced in the previous chapter. The calculated AP values for each of the classes can be seen in Table 7.

Table 7: Result for the Average Precision (AP) and the mean Average Precision (mAP)

	15 epochs		25 epochs		58 images	
	AP: body and face labeled	AP: only face labeled	AP: body and face labeled	AP: only face labeled	AP: body and face labeled	AP: only face labeled
Conchita	0.414258	--	0.740228	--	0.463219	--
Conchita_face	0.635687	0.712054	0.806642	0.663909	0.43817	0.299242
Eva	0.552716	--	0.622215	--	0.421212	--
Eva_face	0.579799	0.438018	0.559439	0.61915	0.545455	0.363636
Jessy	0.393618	--	0.030303	--	0.212121	--
Jessy_face	0.386281	0.154895	0.116883	0.107045	0.19697	0.511364
Kate	0.512609	--	0.596038	--	0.692646	--
Kate_face	0.283418	0.223759	0.561282	0.425969	0.474026	0.492916
Louis	0.826222	--	0.896973	--	0.671945	--
Louis_face	0.774809	0.781724	0.824241	0.845548	0.712735	0.827273
Manfred	0.761941	--	0.73184	--	0.193915	--
Manfred_face	0.79855	0.779717	0.851747	0.870292	0.695987	0.871146
Maya	0.766382	--	0.80194	--	0.907517	--
Maya_face	0.840642	0.710854	0.835096	0.793153	0.772996	0.935868
Nagano	0.552573	--	0.258953	--	0.631313	--
Nagano_face	0.43175	0.438707	0.416515	0.344658	0.290909	0.121212
Sandra	0.546971	--	0.707739	--	0.163636	--
Sandra_face	0.683925	0.783433	0.845601	0.793895	0.272727	0.595187
Zarah	0.360141	--	0	--	0.090909	--
Zarah_face	0.361472	0.42204	0.090909	0.255947	0	0.181818
mAP	0.573188	0.54452	0.564729	0.571957	0.44242	0.519966

The calculations of the AP value for all classes as well as the mAP give a good overview on the performance of each class as well as the performance of the whole trained network overall. When looking at the values for the individual monkeys it can be seen that the monkeys with higher values have higher values throughout all of the networks while classes with low AP values remain low no matter how the network was trained. These results also show that the use of balanced data sets for training and testing did only slightly benefit the weaker classes but decreased the values for the better classes more than it increased the weaker classes. This can also be seen looking at the mAP which is the lowest for both networks trained with balanced data. To compare the result values for the calculated mAP a good comparison is the

mAP presented by Redmon and Farhadi (2018). In their paper they present the performance of YOLO v3 with a value of 0.579, trained on the COCO data set (Redmon and Farhadi, 2018). Compared to this value, the performance of the network detecting the monkeys is almost the same value. Since the YOLO v3 algorithm is more focused on the speed of the training (Redmon and Farhadi, 2018), the performance of this network can be seen as satisfying. This means that the identification of most of the monkeys works as expected.

The best individual results for the monkeys can be seen when looking at the results of Louis, Manfred and Maya. For those three monkeys the results are good in every training performed. In the detailed table the labels with the highest confidence are always correctly detected labels and those three monkeys are also detected the most. Furthermore, the intersection over union values as well as the precision x recall curve for all of the three monkeys are close to perfection. The values for the Average Precision are the highest out of all of the ten monkeys.

The next best results can be seen when looking at the results of Conchita, Eva, Kate's body and Sandra. The performance of those classes is as good as the overall performance of the training of the settings. This means that those classes are the average, and it can be assumed that if a label is predicted as one of those classes and has a high confidence value, it is very likely that the predicted label identified the monkey correctly. The Average Precision values for those classes are only average because a lot of the ground truthing boxes have not been detected and the recall value is not as high.

The other classes with lower Average Precision values are not detected as good as the already mentioned ones. Most of the times the reason for the misdetection of these classes is the lack of training for those specific monkeys. For Jessy it was mostly the missing training data, while for Zarah the reason for the wrong detections was mostly the similarity to the other monkeys which are preferably detected by the trained network. On the other hand, if Zarah was finally identified, it is most likely that the detection was correct. The surprising results are for Nagano since he is the only young monkey out of the ten individuals. Therefore, it was expected that identifying Nagano would be one of the easiest of the monkeys.

Overall it can be said that identifying the monkeys using YOLO v3 works most of the times. Since the detection for some monkeys worked better as for others, the complete detection is quite balanced. If the prediction has a high confidence value it is most likely that the monkey is detected correctly.

5.4 Results using the new data sets

Since the results presented in the last section have performed as expected, the best result will now be proved using the acquired data throughout this thesis. The time skip from the used data to the newly acquired data is over a year so the monkeys will not look exactly the same as in the original data set. It will be interesting and important to see if the time skip as well as the morphological changes of the bodies and faces of the monkeys and the seasonal changes will influence the identification of the network. Since the network is trained to analyze the important parts of the monkey to identify them, testing the network with other data will show if the CNN is capable of detecting the monkeys in images it has not only never seen before but in images that are also from different a different year, different angles etc. So basically, the network is now tested using completely new data where the monkeys do not look like they look in the images it was trained with. As an additional challenge, there are three new monkeys the network is not trained to identify. Therefore, these images should not be detected but it can be expected that the network will still detect some labels since it is choosing the label which will look the most similar to those monkeys. This time the network was tested using a confidence threshold of 0.5. This indicates that if a label is detected, the network is quite sure that it detected the correct monkey. An expressive result of this testing run can be seen in Figure 39.



Figure 39: Example of the most results using the newly acquired data

Those two images of Louis, the monkey with the most accurate and most precise identification throughout all of the other testing runs, is not identified at all. This can also be seen when looking at most of the other images of the testing set created with the newly acquired data. The only reasonable results have been accomplished using images out of two videos from

Kate. The detection for Kate within these images worked as expected since she was identified with high confidence. This is also shown in Figure 40 where two images from Kate are displayed.



Figure 40: Good result using the newly acquired data

These two images of Kate show that not all monkeys were detected wrong. Nevertheless, those correct detections do not balance the amount of wrong detections resulting from the testing. The reason for the misdetections is that the network is not capable of detecting the monkeys due to their morphological changes of their seasonal fur. Furthermore, the influence of shadowing in the images, the position of the monkeys and the angles of the acquisition all did not help to correctly identify the monkeys. The amount of correct labels detected is only 62 out of 539 detected labels. This results in an overall precision of 0.11 which means that only one out of ten labels were detected correctly and since most of the correct labels are either Kate or Sandra, such results are not a reasonable detection. This was already expected since the network was mostly trained on those two monkeys and if it has problems detecting the monkeys it is likely to use the classes it was trained for the most.

To summarize the findings testing the network with the newly acquired videos it can be said that the identification of the monkeys did not work most of the cases. Only for two out of five videos from Kate as well as for some images of Sandra the identification worked as expected. This indicates that for detecting and correctly identifying the monkeys throughout the year more data for all the monkeys and throughout different seasons as well as from more angles and more training of the network as well as specialized training is necessary.

6. Discussion

The goal of the work within this project was to train a Convolutional Neural Network using YOLO v3 to correctly identify monkeys of interest. The experiences of this thesis show how the training of a CNN that is capable to detect an animal works, what has to be considered to correctly detect a monkey and which body parts are necessary for the correct detection.

While training the network, the idea was to fasten every process involved to build the final network. This indicates the labeling of the data, the training and testing as well as the evaluation of the network. Especially in the field of preprocessing the data and labeling the images this can be achieved. With the use of Labelbox, the normally time-consuming part of correctly labeling the ground truth boxes for the training of the network can be completed way faster than expected. Since Labelbox as tool is planned to be used by a team, imagining a whole team labeling the incoming data will fasten the whole process tremendously. Furthermore, the process of labeling can be put according to minimum standards so that it can be ensured that all members of the team can process as good as necessary.

For correctly detecting the monkeys within the images, it is necessary to have enough data from all the monkeys respectively. Without enough labeled training data, the network will not be able to correctly identify some of the monkeys due to the lack of training. Another problem arising is the inconsistency of the data as well as the different angles of the acquired data. Since the network is most likely trained to detect a monkey using data from a certain angle if the monkey can be seen from a different angle it can be that the monkey will be identified wrong. To correctly identify the monkeys there is a lot of data covering the different possible positions and angles needed. Additionally, it can be said that, depending on the monkeys, for correctly detecting them it is good to have the face in the image. Although the trained network only detecting the monkeys from behind has performed pretty well, having the face in the image is definitely helping the network when more and more monkeys are added.

The inconsistent data as well as the lack of data are also addressing the biggest problem faced in this thesis: the correct identification of the monkeys over a long period of time but only being trained for a certain time. This issue has to be faced when trying to identify the monkey by just taking an image of the monkey no matter the season nor the background or angle. To solve this problem, the network has to be trained using data of several different times of the year. Only if these things can be solved, the network will be capable of correctly identifying the monkeys with a high probability, high accuracy and a high precision.

7. Conclusion & Summary

The in this thesis covered thematic of automatically identifying monkeys in their natural habitat leads to several conclusions for the overall topic of detecting and identifying monkeys using a Convolutional Neural Network and the YOLO v3 algorithm. The first important outcome of the thesis is that the identification of the Japanese Macaques of interest living at the monkey mountain in Villach, Austria does work with a CNN using YOLO v3. The overall results are as satisfying as they intended to be while the calculation is quite fast compared to other object detection algorithms. All of the tested monkeys deliver one out of two positive results: either the detection of the individual monkey overall is that results with a high value of confidence are correctly detecting the monkey or that the precision of the detected labels is very high which indicates that if the monkey is detected, the prediction of the network is most likely correct. Using those two factors, all of the monkeys have been detected correctly. When addressing physiological and seasonal changes of changes of the monkey, training data for the corresponding changes has to be considered. Without a trained network using images from the same time period as the ones wanted to be detected, the network has struggles to detect the classes it is trained for precisely. Furthermore, the specific challenge with monkeys changing their fur as well as their face colors has to be handled during the training of such a network.

Additionally, it can be concluded that the position of the monkey as well as the angle of the acquisition also have high influence on a correct prediction. Not having all the necessary data is still connected to a lot of troubles and misdetections since the different positions of the monkeys distract the network when only looking at the body morphology. Without the face of the monkey in the image, the algorithm struggles to correctly predict the corresponding class. This is also addressed when experts try to identify the monkeys since they also have to have a look at their face.

Last but not least, another finding of the thesis is that a tailored training for every different kind of analysis when working with a CNN using YOLO v3 is necessary. The in this thesis used implementation as it stands is hard to enroll to another species or any other test object. The algorithm itself is depending a lot on the input data to learn from it. This means that the overall implementation of the animal of interest and the approach can be transferred to other test species or animals while the specific values for certain settings depend a lot on the task for which the network should be trained and therefore have to be adapted accordingly.

The identification of monkeys in their natural habitat with the use of a Convolutional Neural Network and the YOLO v3 algorithm is an interesting approach to detect differences between the individual monkeys and to get to know the monkeys and the specifications a lot better. The in this thesis described literature research is giving an overview about object detection, the detection of monkeys in particular as well as the explanation and understanding of the use of Convolutional Neural Networks as well as the You Only Look Once algorithm and other alternatives. The implementation of this thesis ranges from the preparation for a data acquisition and the factors that have to be taken care of when dealing with Japanese Macaques. Furthermore, the approach is explaining the setting up of an environment to use the YOLO algorithm as well as the environment needed to label monkeys using Labelbox. The methodology is rounded up by the development of an experiment for the proof of the concept, the preprocessing of both the acquired data and the programs needed to automate the process as well as the method for evaluating the performance of the trained CNN.

To implement the necessary data as well as a validation of the research questions, a data acquisition at the monkey mountain in Villach, Austria took place. The acquired data and mostly the original data from last year were used to train and test the CNN using the YOLO v3 algorithm. To achieve good results, a detailed labeling process was done which dealt with each out of the video sequences generated image individually. The algorithm was automatically validated so that the outcomes deliver reasonable results. While training, different parameters for the influence of the training process as well as different data set splits were used. The resulting outcomes of the network were presented as examples as well as evaluated according to a proven evaluation method and were compared to each other and an original value. The precision and the accuracy as well as values like recall (or sensitivity) and the performance were evaluated. Furthermore, the achieved results were tested with different data sets to compare the performance.

The trained network correctly identified all of the monkeys it was trained for, either with a high precision or a high average precision. The additionally used data has demonstrated that without further training and more training data the network is not capable to detect physiological as well as seasonal changes. Furthermore, the detection of the monkeys only using their body works but has to be addressed with a lot of caution. Additionally, identifying the monkeys of interest when more than one monkey is in the image is difficult to achieve. The disturbance of other monkeys strongly influences the performance of the network.

8. Future Work

This chapter will describe the possibilities and opportunities that would be interesting after seeing the results of this thesis. The further investigations after this work can lead to interesting and valuable research. The future work is only mentioning the work which could be directly associated with the work done in this thesis.

8.1 The use of more monkeys for the CNN

The first interesting opportunity would be to further investigate how such a trained Convolutional Neural Network would work increasing the number of monkeys living in the research area. Since the monkey mountain in Villach, Austria is home to over 160 Japanese Macaques, there is still a huge number of monkeys not included in the identification. This leads to opportunities and challenges while adding all of the remaining monkeys to the training of the network. The additional implementation would also raise new questions for the identification of the monkeys. In a lot of cases monkeys who are related to each other do look quite similar, both their body and their face can look quite similar. Especially when it comes to detecting either the mother or the daughter, this is also challenging for the experts working with those monkeys every day.

The addition of more monkeys to the network will come with a lot of challenges and just imagining a data set with over 160 classes, the identification using YOLO v3 will probably reach some limits. Nevertheless, this addition of more monkeys of interest to the training is very important thinking of the overall goal of the monkey mountain to develop an application. This application can be used either for the visitors to get to know the monkeys and also for the researchers to track the monkeys more easily which can help them with their studies as well as to get to know the monkeys and their behavior better.

8.2 Wildlife cameras for data acquisition

The next interesting possibility using the knowledge gathered within this thesis would be to use data which was acquired using wildlife cameras. The training of images or videos from wildlife cameras will add two interesting follow-up questions. The first one addresses the issue of having the face of the monkey in the image to correctly identify the monkey. With the use of wildlife cameras this cannot be assured since it cannot be controlled how the monkeys pass the place where the wildlife camera is mounted. This also indicates how important the

position of such a wildlife camera will be. Depending on the goal of such a project, a CNN trained with the data of a wildlife camera will be an interesting research opportunity at the monkey mountain. Especially since within the last months, the group of monkeys have split up so using such cameras at the right position can provide information on the monkeys.

The second question and also opportunity for the use of wildlife cameras is the fixed position of a camera for acquiring the data used for training and testing. If the network will be capable to detect the monkeys, the video data can be used to do a live detection of the monkeys to also study their behavior while researchers do not have to be around. So the use of wildlife cameras will not only touch different challenges from this project but will also deal with the use of fixed cameras which can be easily enrolled to other test sites.

8.3 Development of an application based on the trained network

Since the network trained in this project detects the monkeys pretty well, the trained network could be used as base for an application for identifying the individual monkeys. This would be very beneficial for the customers in the research area as well as the specialists and students working there. Such an application should use the trained network as a base for information and if an image of the monkey is taken, this image should be tested with the network and should show the result of the identification. Interesting inclusions of such an application would be attributes to the monkeys such as their name, age, gender, and maybe also their relatives. Another good feature of such an application would be the recording of movement of the monkeys. This could only be achieved when the other two mentioned ideas in this chapter would be merged, so that tracking the individual monkeys and their group behavior could be achieved.

To develop such a monkey identification application the network has to be trained so that it identifies all monkeys living in the research area. Especially, when the application will be used to attract customers and give them a better experience, it would be necessary that all monkeys are included. Otherwise, if the application can only detect half the monkeys, the customers will not use the application very long. Since such an application will most likely be developed for smartphones, it is also important that the network will be trained on data with the same resolution as the average smartphone camera. Otherwise the network would for example be able to identify the monkeys using high quality images but not using images from smartphones.

List of Figures

Figure 1: Workflow of the project.....	9
Figure 2: Literature research Mind Map	10
Figure 3: simple neural network; adopted from Maind, 2014.....	14
Figure 4: Simple architecture of a CNN; adopted from O'Shea and Nash, 2015.....	15
Figure 5: Calculation using a convolutional Layer; adopted from Vink, 2017	16
Figure 6: Max-pooling example; adopted from GeeksforGeeks, 2019.....	17
Figure 7: Animal Recognition System; adopted from Trnovszky et al., 2017	19
Figure 8: Importing Keras in TensorFlow, adopted from TensorFlow, n.d.	22
Figure 9: Darknet-53; adopted from Redmon and Farhadi, 2018	23
Figure 10: Method result comparison, adopted from Trnovszky et al., 2017	26
Figure 11: Conceptual Workflow of the research done in this project.....	29
Figure 12: Intersection Over Union; adopted from Padilla et al., 2020,.....	40
Figure 13: Descriptive confusion matrix for evaluation; adopted from Dubey, 2020	41
Figure 14: Formulas Precision and Recall; adopted from Padilla et al., 2020	41
Figure 15: Example Precision x Recall curve; adopted from Padilla et al., 2020	42
Figure 16: Data sets imported into Labelbox	45
Figure 17: Created labels for labeling the monkeys.....	46
Figure 18: Example of a labeled image in Labelbox	47
Figure 19: Script for converting the Labelbox output into the input format.....	51
Figure 20: Example of a good fitting learning curve during the training of a CNN; adopted from Brownlee, 2019	53
Figure 21: Illustration of loss and validation loss during training	55
Figure 22: Illustration of the detection of an image while testing	56
Figure 23: Example of a successful identification	57
Figure 24: Successful identification of the proof of concept (from the left: Eva, Nagano, Sandra	58
Figure 25: Example image: comparison of the different training data sets during proof of concept (from top left to bottom right: only face, only head, only body, all three)	59
Figure 26: Result of the training from behind.....	60
Figure 27: Result using 15 epochs: difference of the confidence (left: only face recognition; right: body and face recognition).....	61

Figure 28: Result using 15 epochs: perfect detection (left: only face recognition; right: body and face recognition)	62
Figure 29: Example: Correct handling of an image without monkeys	62
Figure 30: Recognition of unknown monkeys within the images	63
Figure 31: Difference using 15 and 25 epochs for training (left: 15 epochs, right 25 epochs)	65
Figure 32: Good result using balanced data for training	68
Figure 33: Illustration: cases of True Positive (left), False Positive (middle) and False Negative (right)	71
Figure 34: Addition of the value for Intersection Over Union to the result table	71
Figure 35: Best results of the detection of Louis	72
Figure 36: Last few rows of Louis' result table	73
Figure 37: Precision x Recall curve using the ordered List of labels	73
Figure 38: Precision x Recall curve with the points for the 11-point interpolated evaluation	74
Figure 39: Example of the most results using the newly acquired data	77
Figure 40: Good result using the newly acquired data	78

List of Tables

Table 1: Original monkeys; adapted from Yang, 2020	31
Table 2: Additional captured monkeys; adapted from Yang, 2020	31
Table 3: Amount of newly acquired monkeys.....	44
Table 4: Results using 15 epochs for the training	64
Table 5: Results using 25 epochs for training	66
Table 6: Results using balanced data for the training.....	68
Table 7: Result for the Average Precision (AP) and the mean Average Precision (mAP)	75

References

- Alharbi F., Alharbi A., Kamioka E., 2019. Animal species classification using machine learning techniques. MATEC Web of Conferences 277
- AnalyticsVidhya, n.d. [online]. Available from: <https://www.analyticsvidhya.com/blog/2020/11/how-to-download-install-and-use-nvidia-gpu-for-tensorflow-on-windows/> [Accessed 4th of August 2021].
- Bochkovskiy A., Wang C. Y., Liao H. Y. M., 2020. YOLOv4: Optimal Speed and Accuracy of Object Detection.
- Boesch C. and Lehmann J., 2004. To Fission or to Fusion: Effects of Community Size on Wild Chimpanzee (Pan Troglodytes Verus) Social Organisation. *Behavioral Ecology and Sociobiology*, 56 (3), 207.
- Boesch C., Wittig R. and Deschner T., 2019. The Chimpanzees of the Tai Forest: 40 Years of Research. *Cambridge University Press*.
- Brownlee J., 2018. Difference Between a Batch and an Epoch in a Neural Network [online]. Available from: <https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/> [Accessed 26th August 2021].
- Buehler P., Carroll B., Bhatia A., Gupta V. and Lee D.E., 2019. An automated program to find animals and crop photographs for individual recognition. *Ecological Informatics*, 50(pp. 191-196).
- Chapman C. A. and Wrangham R. W., 1993. Range Use of the Forest Chimpanzees of Kibale: Implications for the Understanding of Chimpanzee Social Organization.
- Chen K., Loy C.C., Gong S. and Xiang T., 2012. Feature mining for localised crowd counting. *BMVC* (Vol. 1, No. 2, p. 3).
- Dalal N. and Triggs B., 2005. Histograms of oriented gradients for human detection. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2005(1). 886-893, doi: 10.1109/CVPR.2005.177.
- Dubey V., 2020. Evaluation Metrics for Object detection algorithms [online]. Available from: <https://medium.com/@vijayshankerdubey550/evaluation-metrics-for-object-detection-algorithms-b0d6489879f3> [Accessed 26th August 2021].
- Emery Thompson M., Muller M., Machanda Z., Otali E. and Wrangham R. W., in press. The Kibale Chimpanzee Project: Over Thirty Years of Research, Conservation, and Change. *Biological Conservation*.

Fileinfo, n.d. [online] Available at: <https://fileinfo.com/extension/mts> [Accessed 6th of August 2021].

Fortson, L., Masters, K., Nichol, R., Borne, K., Edmondson, E., Lintott, C., ... Wallin, J. (2012). Galaxy zoo: Morphological classification and citizen science. In M. J. Way, J. D. Scargle, K. M. Ali, & A. N. Srivastava (Eds.), *Advances in machine learning and data mining for astronomy* (pp. 213– 236). Boca Raton, FL: CRC Press.

Gandhi R., 2018. R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms [online]. Available from: <https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e> [Accessed 22th October 2020].

GeeksForGeeks, 2019. CNN | Introduction to Pooling Layer. [Online] Available at: <https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/> [Accessed 20th January 2021].

GeeksForGeeks, 2020. Choose optimal number of epochs to train a neural network in Keras [online]. Available at: <https://www.geeksforgeeks.org/choose-optimal-number-of-epochs-to-train-a-neural-network-in-keras/> [Accessed 23rd August 2021].

Girshick R., Donahue J., Darrell T. and Malik J., 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 580-587).

Girshick R., 2015. Fast R-CNN. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 1440-1448

Goodall J., 1986. *The Chimpanzees of Gombe : Patterns of Behavior*. Cambridge, Mass. : Belknap Press of Harvard University Press.

Gopika P., Krishnendu C.S., Hari Chandana M., Ananthakrishnan S., Sowmya V., Gopalakrishnan E.A., Soman K.P., 2020. Single-layer convolution neural network for cardiac disease classification using electrocardiogram signals. *Deep Learning for Data Analytics*, 21-35.

Guresen E. and Kayakutlu G., 2011. Definition of artificial neural networks with comparison to other networks. *Procedia Computer Science*, 3, 426-433

Haykin S., 2009. *Neural Networks and Learning Machines*. Third Edition

Heinicke S., Mundry R., Boesch C., Hockings K. J., Kormos R., Ibnou Ndiaye P., Tweh C. G., Williamson E. A. and Kühl H. S., 2019. *Towards Systematic and Evidence-Based Conservation*

Planning for Western Chimpanzees. *American Journal of Primatology*, 81(9). e23042. <https://doi.org/10.1002/ajp.23042>.

Howard A.G., Zhu M., Chen B., Kalenichenko D., Wang W., Weyand T., Andreetto M. and Adam H., 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

Humle T., Colin C., Laurans M. and Raballand E., 2011. Group Release of Sanctuary Chimpanzees (Pan Troglodytes) in the Haut Niger National Park, Guinea, West Africa: Ranging Patterns and Lessons So Far. *International Journal of Primatology*, 32(2), 456–473. <https://doi.org/10.1007/s10764-010-9482-7>.

ImageNet, n.d. [online]. Available from: <https://www.image-net.org/about.php> [Accessed 26th August 2021].

Kaur T., Singh J., Tong S., Humphrey C., Clevenger D., Tan W. and Szekely B., 2008. Descriptive Epidemiology of Fatal Respiratory Outbreaks and Detection of a Human-Related Metapneumovirus in Wild Chimpanzees (Pan Troglodytes) at Mahale Mountains National Park, Western Tanzania. *American Journal of Primatology*, 70(8). 755–765. <https://doi.org/10.1002/ajp.20565>.

Muehleemann A., 2019. TrainYourOwnYOLO: Building a Custom Object Detector from Scratch [online]. Available from: <https://github.com/AntonMu/TrainYourOwnYOLO> [Accessed 4th of August 2021].

Keras, 2021. [online]. Available from: <https://keras.io/> [Accessed 22th January 2021].

Krogh A., 2008. What are artificial neural networks?. *Nature Biotechnology*. Volume 26. 195-197

Labelbox, n.d. [online]. Available from: <https://labelbox.com/product/platform> [Accessed 22th October 2020].

LeCun Y., Bengio Y., Hilton G., 2015. Deep Learning. *Nature* 521. 436-444

Li J., Gu J., Huang Z., Wen J., 2019. Application Research of Improved YOLO V3 Algorithm in PCB Electronic Component Detection. *Applied Sciences* 2019. 9, 3750.

Lonsdorf E., Travis D., Ssuna R., Lantz E., Wilson M., Gamble K. and Terio, K., 2014. Field immobilization for treatment of an unknown illness in a wild chimpanzee (pan troglodytes schweinfurthii) at gombe national park, tanzania: findings, challenges and lessons learned. *Journal of Primatology*, 55(1). <https://doi.org/10.1007/s10329-013-0372-4>.

Lowe D.G., 1999. Object recognition from local scale-invariant features. *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference*. 2. 1015-1025

Maind S. B., Wankar P., 2014. Research Paper on Basic of Artificial Neural Network. *International Journal on Recent and Innovation Trends in Computing and Communication*, Volume 2, 96-100.

Mitani J. C., 2019. Simultaneous Outbreaks of Respiratory Disease in Wild Chimpanzees Caused by Distinct Viruses of Human Origin. *Emerging Microbes & Infections*, 8(1). 139–49. <https://doi.org/10.1080/22221751.2018.1563456>.

Negrey J. D., Reddy R. B., Scully E. J., Phillips-Garcia S., Owens L. A., Langergraber K. E. and Nishida T., 1968. The Social Group of Wild Chimpanzees in the Mahali Mountains. *Primates*, 9(3), 167–224. <https://doi.org/10.1007/BF01730971>.

O'Shea K. and Nash R., 2015. An Introduction to Convolutional Neural Network

Padilla R., Netto S. L., da Silva E. A. B., 2020. A Survey on Performance Metrics for Object-Detection Algorithms. *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, 237-242 (2000).

Pavlíček J., Jarolímek J., Jarolímek J., Pavlíčková P., Dvořák S., Pavlík J. and Hanzlík P., 2018. Automated Wildlife Recognition. *Agris on-line Papers in Economics and Informatics*, X, 51-60.

Paoletti M.E., Haut J.M., Plaza J., Plaza A., 2018. A new deep convolutional neural network for fast hyperspectral image classification. *ISPRS Journal of Photogrammetry and Remote Sensing*. 145. 120-147

Plumptre A. J., 2010. Eastern Chimpanzee (*Pan Troglodytes Schweinfurthii*), Status Survey and Conservation Action Plan, 2010-2020, 56.

Ranjbar S., Moghadas Nejad F., Zakeri H., Gandomi A. H., 2020. Computational intelligence for modelling of asphalt pavement surface distress. *New Materials in Civil Engineering*, 79-116

Ren S., He K., Girshick R., Sun J., 2016. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. arXiv: 1506.01497v3

Redmon J., Divvali S., Girshick R., Farhadi A., 2016. You Only Look Once: Unified, Real-Time Object Detection.

Redmon J. and Farhadi A., 2017. YOLO9000: Better, Faster, Stronger. arXiv: 1612.08242

Redmon J. and Farhadi, A., 2018. YoloV3: An incremental improvement. arXiv 2018. arXiv preprint arXiv:1804.02767.

Shah, T., 2017. About Train, Validation and Test Sets in Machine Learning [online]. Available at: <https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7> [Accessed 4th of August 2021].

Shakya S., 2020. Analysis of Artificial Intelligence based Image Classification Techniques. *Journal of Innovative Image Processing*. 2(1). 44-54

Sharma, S., 2017a. Activation Functions in Neural Networks: Sigmoid, tanh, Softmax, ReLU, Leaky ReLU EXPLAINED !!! [Online] Available at: <https://towardsdatascience.com/activation-functions-neural-networks1cbd9f8d91d6> [Accessed 4th of August 2021].

Sharma, S., 2017b. Epoch vs Batch Size vs Iterations [online]. Available at: <https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9> [Accessed 18th of August 2021].

Stevens E., Antiga L., Viehmann T., 2020. *Deep Learning with PyTorch*. Manning Publications Co.

Swanson, A. A., Kosmala, M., Lintott, C. C., Simpson, R. R., Smith, A., & Packer, C. (2015). Snapshot Serengeti, high-frequency annotated camera trap images of 40 mammalian species in an African savanna. *Scientific Data*, 2,

Techradar, 2019. What is a neural network? [online], Available from: <https://www.techradar.com/news/what-is-a-neural-network> [Accessed 22th October 2020].

TesnorFlow, n.d. [online], Available from: <https://www.tensorflow.org/> [Accessed 28th January 2021].

Terra J., 2021. Keras vs Tensorflow vs Pytorch: Understanding the Most Popular Deep Learning Frameworks. [online] Available from: <https://www.simplilearn.com/keras-vs-tensorflow-vs-pytorch-article> [Accessed 8th February 2021].

Trnovskiy T., Kamencay P., Orjesek R., Benco M., Sykora P., 2017. Animal Recognition System Based on Convolutional Neural Network. *Digital image processing and computer graphics*. 15(3). 517-525.

van Gemert J.C., Verschoor C.R., Mettes P., Epema K., Koh L.P. and Wich S., 2014. Nature conservation drones for automatic localization and counting of animals. *European Conference on Computer Vision* (pp. 255-270). Springer.

Vasuka A. and Govindaraju S., 2017. *Deep Neural Networks for Image Classification*. In: *Deep Learning for Image Processing Applications*. IOS Press

- Vink R., 2017. Deep learning music classifier part 2. Computer says no!. [online] Available at: <https://www.ritchievink.com/blog/2017/06/04/deep-learning-music-classifierpart-2.-computer-says-no/> [Accessed 8th February 2021].
- Willi M., Pitman R. T., Cardoso A. W., Locke C., Swanson A., Boyer A., Veldthuis M., Fortson L., 2018. Identifying animal species in camera trap images using deep learning and citizen science. *Ecology and Evolution*. 10(1). 80-91
- Wrangham, R. W. 1975. *Behavioural Ecology of Chimpanzees in Gombe National Park, Tanzania*. Thesis, University of Cambridge. <https://doi.org/10.17863/CAM.16415>.
- Yamashita R., Nishio M., Kinoshita K., Togashi K., 2018. Convolutional neural networks: an overview and application in radiology. *Insights into Imaging*, 9, 611-629
- Yang, J., 2020. *Identifizierung von Affen anhand von Videos - Beispiel Affenberg in Landskron*. Unpublished Bsc. Thesis, Geoinformation and Environmental Technologies, Carinthia University of Applied Sciences, Villach, Austria
- Zhang Z., Yang L., Zheng Y., 2020. Multimodal medical volumes translation and segmentation with generative adversarial network. *Handbook of Medical Image Computing and Computer Assisted Intervention*. 183-204
- Zhu W., Ma Y., Zhou Y., Benton M., Romagnoli J., 2018. Deep Learning Based Soft Sensor and Its Application on a Pyrolysis Reactor for Compositions Predictions of Gas Phase Components. *Computer Aided Chemical Engineering*, 44, 2245-2250
- Zhu R., Tu X. and Xiangji Huang J., 2020. Deep learning on information retrieval and its applications. *Deep Learning for Data Analytics 2020*. 125-153
- Zooniverse, n.d. [online] Available at: <https://www.zooniverse.org/about> [Accessed 8th February 2021].

Appendix A

One output label from Labelbox:

```
{ "ID": "ckrj87hziggg70y8u437f7fpu", "DataRowID": "ckrj7rxiedwjm0yp2fzsd4am", "LabeledData": "https://storage.labelbox.com/cjxwlpgi22v080794sa41163t%2F2b23e660-ef4f-d4ca-5cb8-42b5b9716a87-Conchita_307801.jpg?Expires=1629046832437&KeyName=labelbox-assets-key-3&Signature=XiTxB1jRJfKQb8XJ-o5okHZ72Ds", "Label": { "objects": [ { "featureId": "ckrj87q0t002l266dyilz584v", "schemaId": "ckrj82w3xwv4y0y8d2eciexco", "color": "#1CE6FF", "title": "Conchita", "value": "conchita", "bbox": { "top": 165, "left": 963, "height": 834, "width": 622 }, "instanceURI": "https://api.labelbox.com/masks/feature/ckrj87q0t002l266dyilz584v?token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOiJja2xjZ2g5ZzJqMjZwMDczOTc4bWYybWZtIiwib3JnYW5pemF0aW9uSWQiOiJjanh3MXBnaTIydjA4Mdc5NHhNDFsNjN0IiwiaWF0IjoxNjI3ODM3MjMxLCJleHAiOiJlE2MzA0MjkyMzF9.WpgSWqYRgEmUWXY05pUNQw-XVBNfwha9FQ585ou-XdA" }, { "featureId": "ckrj87tci002o266dyx6ipuyh", "schemaId": "ckrj82w3yww500y8dakyqbks", "color": "#FF34FF", "title": "Conchita_face", "value": "conchita_face", "bbox": { "top": 332, "left": 1330, "height": 189, "width": 116 }, "instanceURI": "https://api.labelbox.com/masks/feature/ckrj87tci002o266dyx6ipuyh?token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOiJja2xjZ2g5ZzJqMjZwMDczOTc4bWYybWZtIiwib3JnYW5pemF0aW9uSWQiOiJjanh3MXBnaTIydjA4Mdc5NHhNDFsNjN0IiwiaWF0IjoxNjI3ODM3MjMxLCJleHAiOiJlE2MzA0MjkyMzF9.WpgSWqYRgEmUWXY05pUNQw-XVBNfwha9FQ585ou-XdA" } ], "classifications": [ ] }, "CreatedBy": "lukas.schaffhauser@edu.fh-kaernten.ac.at", "ProjectName": "monkey_mountain", "Created At": "2021-07-25T13:17:48.000Z", "UpdatedAt": "2021-07-25T13:17:48.356Z", "Seconds to Label": 2.966, "ExternalID": "Conchita_307801.jpg", "Agreement": -1, "Benchmark Agreement": -1, "Benchmark ID": null, "Dataset Name": "Conchita_30", "Reviews": [ ], "ViewLabel": "https://editor.labelbox.com?project=ckrj826upieum0y8uak5df4m7&label=ckrj87hziggg70y8u437f7fpu", "Has Open Issues": 0, "Skipped": false },
```

Output from Labelbox of a skipped image:

```
{ "ID": "ckrs4vdli4e0j0y990qgi6vja", "DataRowID": "ckrs4okex07gb0zvr49nq28xa", "LabeledData": "https://storage.labelbox.com/cjxwlpgi22v080794sa41163t%2F1a62d823-4cb6-b9bb-a5c1-f41e9cc00338-Maya00721.jpg?Expires=1629046833852&KeyName=labelbox-assets-key-3&Signature=kS1g5XZcRlZCeRGqSfXJNah7Mx0", "Label": { }, "CreatedBy": "lukas.schaffhauser@edu.fh-kaernten.ac.at", "ProjectName": "monkey_mountain", "Created At": "2021-07-31T18:54:12.000Z", "UpdatedAt": "2021-07-31T18:54:23.000Z", "Seconds to Label": 5.539, "ExternalID": "Maya00721.jpg", "Agreement": -1, "Benchmark Agreement": -1, "Benchmark ID": null, "Dataset Name": "Maya_30", "Reviews": [ ], "ViewLabel": "https://editor.labelbox.com?project=ckrj826upieum0y8uak5df4m7&label=ckrs4vdli4e0j0y990qgi6vja", "Has Open Issues": 0, "Skipped": true },
```

Appendix B

```
import pandas as pd
import json
import csv
import re
import numpy as np

with open(r'C:\Users\Benutzer1\Desktop\2nd_try\Test_Labels_all_10.csv') as e:
    refer = pd.read_csv(e, sep = ";")

with
open(r'C:\Users\Benutzer1\Desktop\2nd_try\TrainYourOwnYOLO\Data\Source_Images\58_all_monkey_detection\58_all_monkey_detection.csv') as f:
    #with
    open(r'C:\Users\Benutzer1\Desktop\2nd_try\TrainYourOwnYOLO\Data\Source_Images\Test_Image_Detection_Results_all10_final\Detection_Results_all10_final.csv') as f:
    #with
    open(r'C:\Users\Benutzer1\Desktop\2nd_try\TrainYourOwnYOLO\Data\Source_Images\new_all_monkey_detection\new_all_monkey_detection.csv') as f:
        data = pd.read_csv(f, sep = ";")

gt = pd.DataFrame(refer)
pb = pd.DataFrame(data)

im_name_pb = pb['image']
im_name_gt = gt['image']
gt_labl = gt['number']
pb_labl = pb['label']

def calculate_IoU(box_A, box_B):

    # get the bounding box information/(x,y) coordinates of the
    intersection rectangle
    x_min = max(gt_box[0], pred_box[0])
    y_min = max(gt_box[1], pred_box[1])
    x_max = min(gt_box[2], pred_box[2])
    y_max = min(gt_box[3], pred_box[3])

    # compute the area of overlap
    area_of_overlap = max(0, x_max - x_min + 1) * max(0, y_max - y_min + 1)

    # compute the area of both the prediction and ground-truth bounding
    boxes
    gt_box_area = (gt_box[2] - gt_box[0] + 1) * (gt_box[3] - gt_box[1] + 1)
    pred_box_area = (pred_box[2] - pred_box[0] + 1) * (pred_box[3] -
pred_box[1] + 1)

    # compute the Intersection Over Union
    intersection_over_union = area_of_overlap / \
        float(gt_box_area + pred_box_area - area_of_overlap)

df_IoU = pd.DataFrame(columns= ['image', 'label',
'label_number', 'confidence', 'IoU'])
```



```

#restart = False
prove_list = []

for i in range(len(pb)):
    for j in range(len(gt)):

        prove = im_name_pb[i]
        ueber = gt[gt['image'].str.contains(prove)==True]

        if im_name_pb[i] == im_name_gt[j]:

            if j in prove_list: j+1
            im_name = im_name_pb[i]
            ref_ind = pb_labl[i]

            if pb_labl[i] == gt_labl[j]: #correct detection
                ln = gt['label'][j]

                prove_list.append(j)

                gt_box =
[gt['xmin'][j],gt['ymin'][j],gt['xmax'][j],gt['ymax'][j]]
                pb_box =
[pb['xmin'][i],pb['ymin'][i],pb['xmax'][i],pb['ymax'][i]]

                IoU = calculate_IoU(gt_box,pb_box)

                df_IoU = df_IoU.append(
                    {
                        'image': im_name,
                        'label': ln,
                        'label_number': ref_ind,
                        'confidence': pb['confidence'][i],
                        'IoU': IoU
                    }, ignore_index=True
                )
                break

            ln = pb['label_w'][i]
            print(ln, im_name)
            naga = "exit"
            monk = "exit"

            if ref_ind == 0 or ref_ind == 1:
                monk = "eva"

            if ref_ind == 2 or ref_ind == 3:
                monk = "Nagano"
                naga = "C003"

            if ref_ind == 4 or ref_ind == 5:
                monk = "Sandra"
            if ref_ind == 6 or ref_ind == 7:
                monk = "Conchita"
            if ref_ind == 8 or ref_ind == 9:
                monk = "Jessy"
            if ref_ind == 10 or ref_ind == 11:
                monk = "Kate"
            if ref_ind == 12 or ref_ind == 13:
                monk = "Louis"
            if ref_ind == 14 or ref_ind == 15:

```

```

        monk = "Manfred"
    if ref_ind == 16 or ref_ind == 17:
        monk = "Maya"
    if ref_ind == 18 or ref_ind == 19:
        monk = "Zarah"

    if monk in im_name or naga in im_name:

        j+1
        if j in prove_list:

            j+1

    else: # a label was detected but is not in the corresponding
image

        df_IoU = df_IoU.append(
            {
                'image': im_name,
                'label': ln,
                'label_number': ref_ind,
                'confidence': pb['confidence'][i],
                'IoU': -1
            }, ignore_index=True
        )
        break

    elif ueber.empty: #a label was detected but the ground truth box
was not labeled

        df_IoU = df_IoU.append(
            {
                'image': prove,
                'label': pb['label_w'][i],
                'label_number': pb_labl[i],
                'confidence': pb['confidence'][i],
                'IoU': -2
            }, ignore_index=True
        )

        break

    j+1

df_IoU.to_csv(r'C:\Users\Benutzer1\Desktop\2nd_try\58_IoU_combined_all_monk
eys.csv', sep=',', index=False)

```