FACULTY OF BIOSCIENCE ENGINEERING

AUTOMATED RECOGNITION OF PEOPLE AND IDENTIFICATION OF ANIMAL SPECIES IN CAMERA TRAP IMAGES

word count: 22348

Laura Hoebeke
Student ID: 01302335

Promotors: prof. dr. Bernard De Baets, dr. ir. Michiel Stock
Tutor: dr. ir. Stijn Van Hoey

Master thesis submitted for obtaining the degree: Master of Science in Bioscience Engineering: Land and Water Management

Academic year: 2017 - 2018

GHENT UNIVERSITY

Ghent, 8 June 2018

The promoters,

prof. dr. Bernard De Baets

dr. ir. Michiel Stock

The tutor,

The author,

dr. ir. Stijn Van Hoey

Laura Hoebeke

DANKWOORD

Eerst en vooral wil ik graag mijn promotoren en tutor bedanken. Zij hebben de basis gelegd van een vruchtbare samenwerking met het Instituut voor Natuur- en Bosonderzoek waardoor ik de kans kreeg aan dit interessante onderwerp te werken tijdens het laatste jaar van mijn opleiding tot bio-ingenieur. Mede dankzij hun begeleiding en advies heb ik mijn masterproef tot een goed einde kunnen brengen. In het bijzonder wil ik graag Michiel bedanken voor de vele tijd die hij in mij geïnvesteerd heeft om mijn werk wekelijks op te volgen en bij te sturen waar nodig.

Stijn en Jim van INBO hebben er voor gezorgd dat ik over de nodige cameravalbeelden en bijbehorende informatie beschikte. Ik kon ook altijd bij hen terecht wanneer ik met vragen zat over cameravallen of over de beelden. Zonder geannoteerde beelden was deze masterproef niet mogelijk geweest. Het spreekt dan ook voor zich dat iedereen die hieraan meegewerkt heeft een vermelding verdient in dit dankwoord. In het bijzonder wil ik hiervoor Jolien bedanken die het grootste deel van de data geannoteerd heeft. Werken met een groot volume aan data bracht ook enkele technische complicaties met zich mee. Ik wil dan ook graag Jan bedankten om mij hierin bij te staan. Stijn en Jim wil ik bedanken om mij op weg te helpen met het gebruik van de GPU, een onmisbaar onderdeel voor het trainen van het neuraal netwerk. Joris verdient hier ook zeker een vermelding aangezien hij altijd klaar stond om problemen met GitHub en LaTeX te verhelpen. Bij technische problemen met Python kon ik altijd rekenen op de hulp van mijn broer, Matthias. Daarnaast heeft hij ook meermaals zijn ideeën gedeeld, gevraag en ongevraagd, tijdens gesprekken over mijn masterproef.

Ten slotte wil ik ook heel graag mijn ouders bedankten. Zij hebben mij de mogelijkheid geboden om een universitaire opleiding te volgen en hebben mij hierin onvoorwaardelijk gesteund waardoor ik zonder veel problemen het eindpunt heb kunnen bereiken.

Laura Hoebeke

Juni 2018

SAMENVATTING

Cameravallen worden steeds vaker gebruikt om dieren in het wild te monitoren. Het grote voordeel van cameravallen in vergelijking met andere methoden is dat zeer accurate data verkregen kan worden zonder dat de dieren verstoord moeten worden door ze bijvoorbeeld te merken of een halsband aan te doen om hun positie te kunnen volgen. Bovendien kan de data ook verzameld worden zonder dat er iemand aanwezig moet zijn om te observeren. Cameravalprojecten produceren echter grote hoeveelheden data die vaak nog manueel verwerkt worden. Convolutionele neurale netwerken kunnen aangewend worden om dit arbeidsintensieve proces grotendeels te automatiseren.

In deze masterthesis worden bestaande, handmatig gelabelde beelden van een cameravalstudie uitgevoerd door het Instituut voor Natuur- en Bosonderzoek in samenwerking met de Universiteit Hasselt gebruikt om een convolutioneel neuraal netwerk te trainen om de beelden hiërarchisch te classificeren. Op deze manier kunnen cameravalbeelden automatisch gelabeld worden of het netwerk kan geïntegreerd worden in annotatietoepassingen om een suggestie te geven aan de gebruikers en zo het annotatieproces te versnellen.

Naast het bevestigen van de aanwezigheid van diersoorten kunnen de beelden ook andere nuttige informatie bevatten, zoals de eigenschappen en het gedrag van een dier. Daarom kan het nuttig zijn om de hulp in te roepen van vrijwilligers via *citizen science* om de grote hoeveelheden cameravalbeelden te verwerken. Omdat de camera's ook in openbare gebieden zoals natuurreservaten geplaatst worden, kan het zijn dat er toevallige voorbijgangers op de cameravalbeelden staan. Omwille van privacyredenen kunnen beelden die mensen bevatten echter niet openbaar worden gemaakt. Daarom zal het neuraal netwerk ook getraind worden om mensen te herkennen zodat deze beelden automatisch uit de dataset verwijderd kunnen worden. Hierna kunnen de beelden dan beschikbaar gesteld worden aan vrijwilligers.

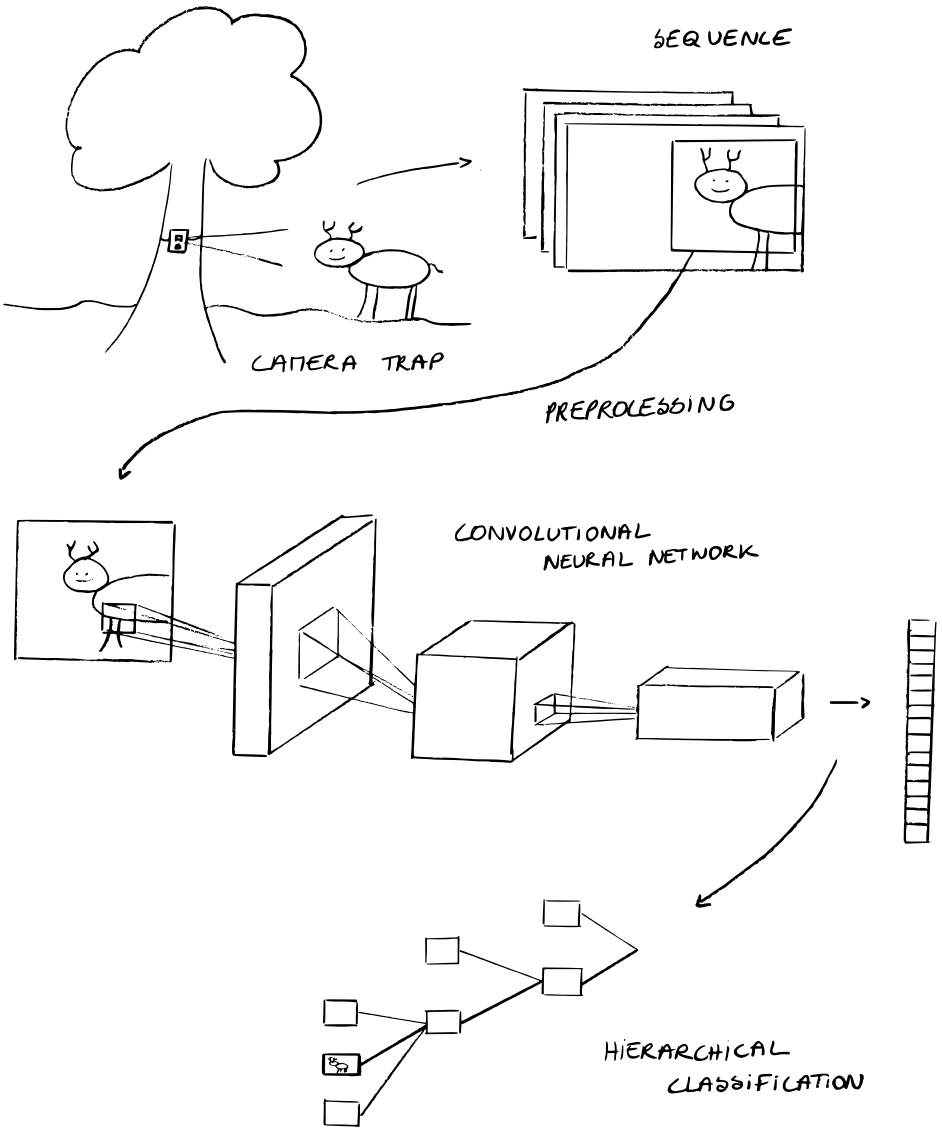
SUMMARY

Camera traps are increasingly being used in wildlife monitoring. The great advantage of camera traps in comparison with other sampling methods is that very accurate data can be collected without the animal being collared or tagged nor the researcher being present. However, such camera trapping frameworks produce high volumes of pictures which often need to be reviewed manually. Convolutional neural networks can be used to automate this labour intensive process.

In this master thesis, existing manually labelled images from a camera trap study conducted by the Research Institute for Nature and Forest in collaboration with Hasselt University are used to train a convolutional neural network to hierarchically classify animal species. In this way, images can be automatically labelled or the network can be incorporated into annotation applications to provide a suggestion to the users and as such speed up the annotation process.

In addition to conveying the presence or absence of species, the images may contain other useful information, for example animal attributes and behaviour. Therefore, getting help from wildlife enthusiasts via citizen science, may be desirable to review the large amounts of data. However, since cameras are mounted in public spaces such as nature reserves, there always exists the risk that passers-by have triggered the camera traps. For privacy reasons, images showing people cannot be made public. Therefore, the neural network will also be trained to recognize people to be able to automatically remove these images from the dataset. After removing these images, the data can be made available to volunteers.

GRAPHICAL ABSTRACT



CONTENTS

CHAPTER 1

INTRODUCTION

Camera traps are increasingly being used in wildlife monitoring. The great advantage of camera trapping compared to other sampling methods such as direct observation, trapping and radio tracking, is that very accurate data can be collected without the animal being collared or tagged nor the researcher being present. By using camera traps the cost, labour and logistics of observation can be reduced. To review the camera trap images, researchers can get help from wildlife enthusiasts via citizen science. Volunteers from the general public annotate the camera trap images, usually via online citizen science platforms.

1.1 Problem statement

Camera trapping frameworks produce large numbers of images which often need to be reviewed manually. This labour-intensive process is a limiting factor in the use of camera traps. Camera trapping has greatly increased sampling, but analysing the images remains a bottleneck in turning the data into information on animal presence, abundance and behaviour.

The use of citizen science is limited by the fact that for privacy reasons, images showing people cannot be made public. When camera traps are mounted in public spaces such as nature reserves, there always exists the risk that passers-by have triggered the camera. The data therefore needs to be reviewed and images showing people need to be removed, before the remaining images can be made available to volunteers.

1.2 Objectives of this research

The objective of this master thesis is to build an image classification model to speed up the annotation process of camera trap images. The resulting model can be used to automatically classify images or to provide suggestions to researchers or volunteers. Manually labelled images from an ongoing camera trap survey from the Research Institute for Nature and Forest (INBO) will be used to train a convolutional neural network. Firstly, the number of images that needs to be review manually can substantially be reduced by training

a model to detect empty images. Furthermore, the network will be trained to identify the animal species in the images. As citizen science can provide other useful information such as animal attributes and behaviour, the network will also be trained to recognise humans. In this way, images containing humans can be automatically removed from the dataset, after which the data can be made available to the general public.

1.3 Outline of this dissertation

Chapter 2 contains an introduction to camera trapping for wildlife monitoring. The general properties of camera traps are explained and a non-exhaustive overview is given of possible applications of camera trap images. In this chapter, also citizen science is touched upon and examples of successful citizen science project are given. Chapter 3 gives an overview of the available data. The structure of the camera trap images and their associated labels is described. At the end of this chapter, an overview can be found of the observed bird and mammal species. Chapter 4 starts with a general introduction on image classification after which challenges inherent to the classification of camera images are discussed. In Chapter 5, the preprocessing steps, applied to extract the regions of interest from the camera trap images, are described. Chapter 6 gives a brief introduction to deep learning after which the properties of neural networks, in particular convolutional neural networks, are discussed. The classification tree used to hierarchically classify the images and more information on this method can be found at the end of Chapter 6. In Chapter 7, the different steps taken during training of the convolutional neural network are described. The performance of the network is evaluated using confusion matrices. The chapter ends with a visualisation of different elements learnt by the neural network. This allows us to get a better understanding of the internal operations and the behaviour of convolutional neural networks.

The code is published via Zenodo[1]. This includes a tutorial on how to use the trained network to classify new sequences and code to retrain the neural network itself.

[1] DOI: 10.5281/zenodo.1262684

CHAPTER 2

CAMERA TRAPS FOR WILDLIFE MONITORING

Camera traps are increasingly being used to study wildlife. This comes with some technical and methodological challenges. In this chapter, the general properties of camera traps are explained as well as their advantages and disadvantages. A non-exhaustive overview is given of possible applications of camera trap images. Lastly, citizen science is touched upon, which can be a helpful tool to review the large number of images camera trapping surveys generate.

2.1 A brief history of camera trapping

Already in 1890 George Shiras developed a method using a tripwire and a flash system allowing that wild animals be auto-photographed without disturbing them (Gomez et al., 2017). Since the early 20th century, camera trapping of wildlife has been in practice (Rowcliffe and Carbone, 2008). However, due to technical challenges and high costs, camera traps did not become popular until the development of commercial camera systems in the late 20th century (Swann et al., 2004). Nowadays, more complex systems are available, using infrared beams as a triggering device. The modern camera traps are small, portable, resistant to rough weather conditions and much more affordable (Gomez et al., 2017). As a result, they are becoming a mainstream tool for wildlife monitoring (Rowcliffe and Carbone, 2008).

2.2 Passive infrared motion detector

Passive infrared sensors are nowadays the most commonly used type of sensor in camera traps (Swann et al., 2011). They detect differences between the background temperature

Figure 2.1: Reconyx HyperFire camera. Image from Reconyx (2013).

and the surface temperature of objects. The camera is triggered when a rapid change in temperature occurs (Welbourne et al., 2016).

Infrared-triggered cameras are prone to two types of errors: false triggers (false positives) and failure to photograph the animal (false negatives) (Swann et al., 2004). Thermal radiation from non-target objects and the sensor itself causes background noise. Therefore, a threshold is set to prevent false triggers (Swann et al., 2004). Nevertheless, in practice false triggers still occur. Possible causes are wind or rain moving either vegetation or the support of the camera, moving water, radiant heat in only a portion of the detection zone or sunlight reflection (Swann et al., 2004). Also an animal that is within the detection zone but outside the camera's range or a time lag between the passing of an animal and the starting of the camera causes false triggers (Swinnen et al., 2014). The background environment consists of a thermally heterogeneous configuration of objects such as trees, shrubs and grasses. These elements have different surface temperatures due to their different thermal properties. This thermal heterogeneity of the environment may be a cause of false triggers as moving objects that are warmer or cooler than background substrates, for instance a branch or shrub, can cause the camera to be triggered (Welbourne et al., 2016). The reasons for the second type of error, failure to photograph the animal triggering the camera, are less obvious. In contrast to the false triggers, which are noticed when reviewing the images, there is often no clear explanation of why a unit has failed in the field (Swann et al., 2004). The animal model trials performed by Swann et al. (2004) suggest that cameras may not trigger due to subtle differences in animal speed or height.

2.3 Reconyx HyperFire camera

The images that will be used to train the neural network are taken by Reconyx HyperFire cameras with a passive infrared motion detector, as depicted in Figure 2.1. This detector is

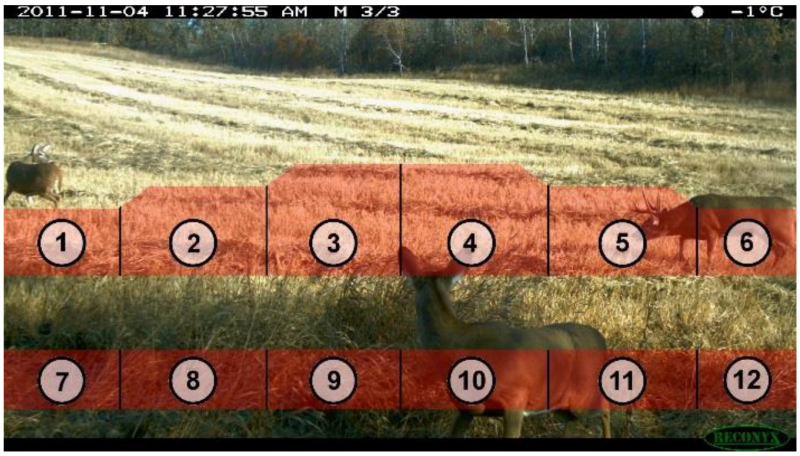


Figure 2.2: Detection bands and zones of the HyperFire motion detector. Image from Reconyx (2013).

aligned with the camera lens so it only detects objects in the field of view of the camera. The motion detector consists of two horizontal detection bands, located one above the other (Figure 2.2). Each band is divided into six zones. According to the manufacturer two things need to happen for the camera to take a picture. Firstly, an object with a temperature different from the background temperature must be present within one of the detection bands. Secondly, that object must be moving into or out of at least one of the twelve motion detection zones (Reconyx, 2013). This means that in Figure 2.2, the deer to the far right and the deer in the middle would trigger the camera, but not the deer to the far left since it is above the detection bands.

The camera takes pictures during the day as well as at night. Switching between day mode and night mode happens automatically. During the day, colour images are taken (Figure 2.3a). At night, monochrome infrared images are taken using the night time infrared illuminator (Figure 2.3b) (Reconyx, 2013).

2.4 Characteristics of camera traps

2.4.1 Advantages

The great advantage of camera traps in comparison with other sampling methods such as direct observation, trapping and tracking, is that very accurate data can be collected without the animal being collared or tagged nor the researcher being present. Camera traps can be used to gain information on highly cryptic species and in difficult terrain where other field methods are likely to fail (Rowcliffe et al., 2008). An additional advantage is that the data can be reviewed by other researchers, unlike data produced by live-trapping or observations (Swann et al., 2011).

(a) Day (b) Night

Figure 2.3: Day mode colour image (a) and night mode monochrome infrared image (b).

Another major benefit is that the images from one study can be used in other studies, even if they do not target the same species (Rowcliffe and Carbone, 2008). However, as stated by many authors (e.g. Rowcliffe and Carbone (2008)), the impressive growth in camera trapping studies is quite uncoordinated. Therefore, to fully exploit their enormous potential, there is a need for greater integration and consensus. Burton et al. (2015), among others, call for more thorough reporting of methodological details.

2.4.2 Disadvantages

In contrast to the advantages of camera traps, which are well-represented in scientific literature, the disadvantages have received less attention (Swann et al., 2011). Most of the issues appear when camera traps are used in the field. One of the most common issues is the loss of data due to equipment failure, for example mechanical problems, improper programming or battery failure (Swann et al., 2004). This is an even bigger issue when using camera traps in remote areas, where it can take months before the failure is noticed (Swann et al., 2011). On the other hand, camera traps mounted in public areas are subject to theft and vandalism (Foster and Harmsen, 2012; Silver et al., 2004). Metal cases and locks can be used to protect the camera against theft and damage by humans or large animals (Rovero et al., 2013), as shown in Figure 2.4. Other issues users of camera traps with passive infrared detectors have to deal with are false triggers and failure to photograph the animal triggering the camera, as mentioned in Section 2.2. False triggers are rarely reported although they give an important indication about the efficiency and expected time effort necessary to process the images (Swinnen et al., 2014).

As for any wildlife survey method, camera trap surveys have to deal with sources of sampling errors such as imperfect detection, where individuals or species present within a sampling area are not always detected (Burton et al., 2015). Since camera traps target mobile objects, they have to contend with imperfect detection at two spatial scales: animals passing through the relatively small camera detection zone may not be detected and animals using some larger area that the camera is assumed to sample may not enter the detection

Figure 2.4: Reconyx HyperFire camera in a security box. Image from Reconyx (2013).

zone (Burton et al., 2015). The probability of detection is affected by many factors operating across different scales, as illustrated by Burton et al. (2015) (see Figure 2.5). This calls for a careful accounting of the relationship between detections and the underlying ecological processes of interest (Burton et al., 2015).

2.5 Research applications of camera trap images

2.5.1 Examples of camera trap applications

Camera trap images have a wide range of possible applications. One of the most straightforward applications is using the images to confirm the presence of a certain animal species. The absence of an animal can not be confirmed using camera traps. When an animal is never captured on an image, this is only weak evidence for its absence and does not necessarily mean that the animal is not present in the area where the camera is located. Henschel et al. (2010) used camera traps to search for lions (*Panthera leo*). This species is listed as vulnerable on the IUCN Red List of Threatened Species and the population is still decreasing. They were able to confirm the presence of *Panthera leo* in two of the lion conservation units surveyed in West Africa, but in none of the units in Central Africa, from which they conclude that their result raises the possibility that no resident lion populations exist in the units in Central Africa.

Other applications of camera traps include simple species inventories, the discovery of new species, abundance estimations, conservation assessments and population dynamics (Rowcliffe and Carbone, 2008). Tobler et al. (2015) for example developed a multi-session multi-species occupancy model to improve estimates for species richness and occupancy for a large data set, based on the total number of detections per species. The model can be used to monitor mammal communities over time and investigate regional and temporal

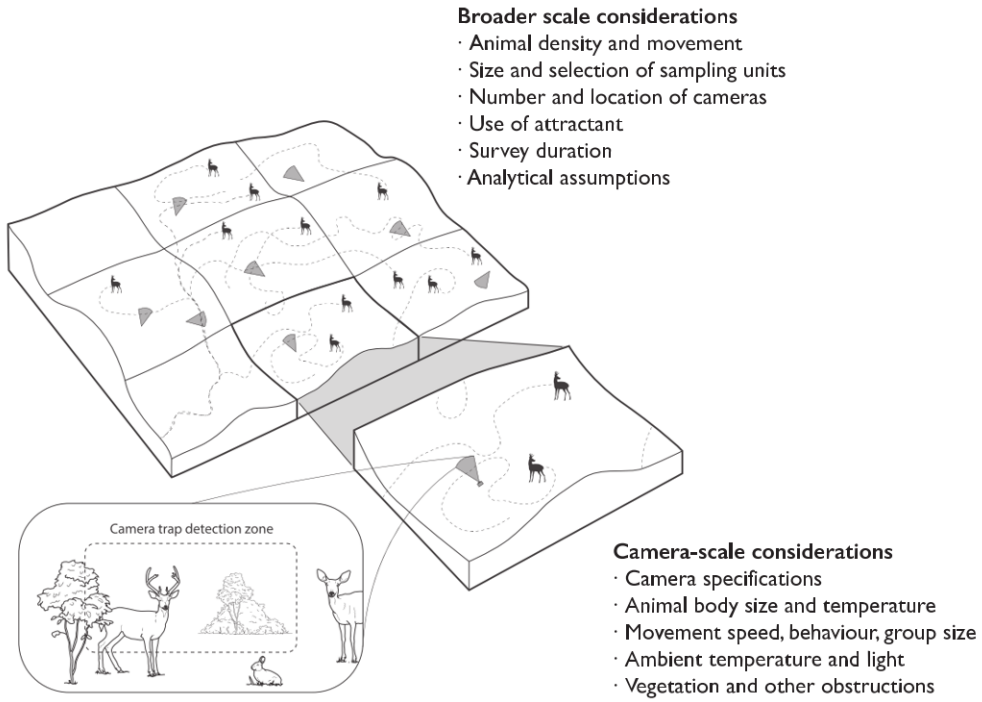


Figure 2.5: Factors affecting detection probability. Image from Burton et al. (2015).

patterns in the distribution and composition of communities and this in relation to natural or anthropogenic factors. Camera trap are also used to try to discover new species. In 2005, a new species of elephant-shrew (*Rhynchocyon*), a small insectivorous mammal, was described. It was discovered in the northern Udzungwa Mountains of Tanzania, based on camera trap images (Rovero et al., 2008). Li et al. (2015) described a newly discovered macaque species from Tibet by means of camera trap images. In southeastern Tibet, macaques live in dense tropical or subtropical evergreen forests where they travel across mountainous terrain. Since it was difficult to follow these unhabituated groups, they opted to use camera traps to be able to compare characteristics of the local macaque populations.

Swinnen et al. (2014) also list studying niche separation, competitive exclusion, population structure, foraging behaviour and biodiversity as possible applications. Survival and recruitment estimations, monitoring populations and communities over time, analysis of habitat associations, studying activity patterns, diet or reproduction, disease monitoring and monitoring of wildlife crossings are mentioned by Rovero et al. (2013). In 2005, the first green bridge in Flanders was constructed in Bierbeek, connecting the eastern and western part of Meerdaalwoud. Figure 2.6 shows an aerial photo of this ecoduct. Camera traps were installed to monitor which animals were using the green bridge. The results showed that most of the target species used the bridge and often even large numbers of them (Lambrechts et al., 2013).

Figure 2.6: Aerial photo of green bridge 'De Warande' in Bierbeek (Flemish Brabant). Image from Lambrechts et al. (2013).

In recent years, camera traps are also increasingly being used in documenting presence of rare species, rare events and rare or melanistic individual animals (Swann et al., 2011). Hedges et al. (2015) discovered a novel modification to infrared flash camera traps, which always forces the camera into night mode. In this way, they were able to consistently and clearly see the spots of melanistic leopards (*Panthera pardus*) in Malaysia (Figure 2.7). This was an important finding since almost all leopards are melanistic in this country, which made it nearly impossible to identify individuals using normal camera traps for estimating leopard density.

2.5.2 Methodological challenges

The above examples show that the possible uses of camera trap image are endless. However, there are still some methodological challenges. As mentioned in Section 2.2, Swann et al. (2004) confirmed the relationship between species body mass and the probability of triggering cameras. From this they conclude that trap rates cannot be used as an index to compare relative abundance across species. There are also factors other than abundance that influence trapping rates, particularly animal movement rates, body size and patterns of habitat use (Rowcliffe and Carbone, 2008). Therefore, early applications mostly relied on individually recognisable animal species, often large cat species (Rowcliffe and Carbone, 2008). Capture-recapture models can be used for species with individually-distinct fur patterns or artificial marks to estimate abundance and density. These models account for the fact that not necessarily all animals in the study area are observed (Rovero et al., 2013). Nowadays, methods are being developed to estimate the abundance and density of species that cannot be individually identified, since the majority of wildlife species are not easily individually identifiable. Rowcliffe and Carbone (2008) state that is must be possible to

Figure 2.7: Example of a photograph of a melanistic leopard taken in daytime colour mode (a) and with the camera forced into using the infrared flash (b). Image from Hedges et al. (2015).

control for the above-mentioned confounding variables and extract the underlying abundance signal in trapping rate data. They proposed a method, the Random Encounter Model (Rowcliffe et al., 2008), which aims to estimate the density of species based on the likelihood that the camera detection zone is crossed by passing animals. Another model was developed by MacKenzie et al. (2002), aiming at occupancy studies. They define occupancy as the proportion of area, patches or sites occupied by a species. Their model estimates site occupancy and detection probability based on repeated presence-absence surveys of multiple sites. Their occupancy estimation accounts for the fact that it is possible that a species is present but not detected.

2.6 Citizen science

To review the large amount of images resulting from camera trapping surveys, it may be helpful to get help from volunteers. Wildlife enthusiasts can help identifying the animals captured on the camera trap images. In addition, the images may contain other useful information, for example animal attributes and behaviour which the volunteers can indicate. The time and effort volunteers invest in these projects is often vital for their success and allows researchers to make meaningful contributions to the expansion of scientific knowledge (Reed et al., 2013). The data obtained through citizen science can also be used to eventually automate this process via machine learning.

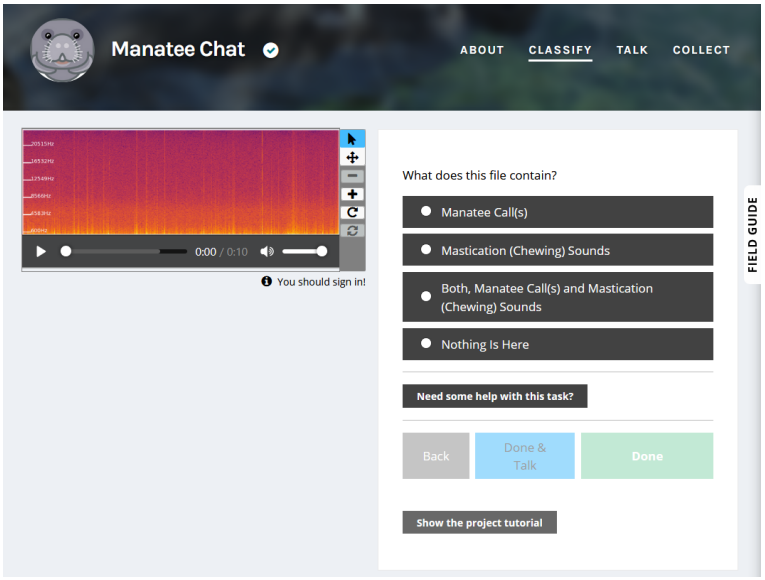


Figure 2.8: The Zooniverse[2] interface for the Manatee Chat project. On the left of the screen, an audio clip is displayed that can be played and on the right the user can select one of the proposed labels.

2.6.1 Virtual citizen science

Originally, citizen science referred to enlisting volunteers from the general public in gathering scientific information themselves (Bonney et al., 2009) or by assisting researchers doing fieldwork (Cohn, 2008). An example of such a project that relies on citizen science is the annual counting of garden birds in Flanders during the last weekend of January, organised by Natuurpunt. In 2018, 33 500 people participated and more than 600 000 birds were recorded (Natuurpunt, 2018). Nowadays, volunteers can make use of the internet to participate in scientific research, resulting in the rise of *virtual citizen science*. This also lowers barriers for volunteers to participate in projects since they only need access to the citizen science website to contribute to projects anywhere in the world (Reed et al., 2013). An example of a successful virtual citizen science project is Foldit[1], where protein folding was turned into a game. In this way, volunteers playing the game discovered for example an ideal protein structure before professional AIDS researchers working on the same problem did (Khatib et al., 2010).

2.6.2 Citizen science platforms

Zooniverse[2] is the world's largest citizen science platform. It hosts a great diversity of projects from various research disciplines such as biology, medicine, literature, arts, astronomy, archaeology and climatology (Reed et al., 2013). Research data is shown to volunteers in the form of images, video and audio. A simple tutorial shows them how to perform the

[1] https://foldit.it
[2] https://www.zooniverse.org

Figure 2.9: The Snapshot Serengeti website interface with all available species options (left) and filters that help narrow users' choices when classifying species (right). Image from Swanson et al. (2015).

required analysis so that they can then identify, classify, mark and label data as researchers would do (Simpson et al., 2014). Figure 2.8 shows the interface of the Manatee Chat project on Zooniverse, where volunteers need to identify and classify manatee calls. A big project that partnered with Zooniverse is Snapshot Serengeti[3]. Figure 2.9 shows the online interface with all available species options and filters that help narrow users' choices when classifying species. In Serengeti National Park (Tanzania), 225 cameras were deployed, which produced 1.2 million sets of pictures. These images were classified by members of the general public, resulting in 10.8 million classifications (Swanson et al., 2015). These results show the power of citizen science platforms. To improve data accuracy, every image was circulated to multiple users after which an algorithm was applied to aggregate the individual classifications into a final 'consensus' dataset. By validating the consensus classifications against images classified by experts, the labelling by volunteers was estimated to be 96.6% accurate (Swanson et al., 2015).

Another upcoming citizen science platform is eMammal[4], a data management system for collecting, organizing and displaying camera trap images and associated metadata collected by volunteers (McShea et al., 2016). In contrast to the Zooniverse projects, it is not possible for volunteers to just classify the camera trap images available on the website. This platform targets studies in which volunteers deploy cameras themselves and upload and process the resulting images. However, on the website there are annotated camera trap images available from projects around the world that one can explore.

[3] https://www.snapshotserengeti.org
[4] https://emammal.si.edu

2.6.3 Motivation of volunteers

Reed et al. (2013) wanted to understand what motivates volunteers to participate in citizen science projects. Therefore, they asked users of Zooniverse to fill out a web survey, assessing various possible motives for their participation. They found that three factors are important: social engagement, interaction with the website and helping. Social engagement refers to the awareness of and interaction with other members of the Zooniverse community. Also the tasks and skills involved motivate the volunteers. The second factor, interaction with the website, comprises awareness, facility and enjoyment from using the various features of Zooniverse projects. The third and last factor, helping, communicates how participants experience positive feelings from helping or volunteering to participate in Zooniverse (Reed et al., 2013). This is important information for scientists who want to start a citizen science project since the success of these research projects often depends on the rate of participation.

2.6.4 Education and awareness

Another development in the application of camera traps is the growing use for education. As mentioned above, citizen science is a great tool for involving the general public in nature research. In this way, also awareness is raised for the projects the images result from (Swann and Perkins, 2014). The citizen science platform eMammal offers the opportunity to schools to deploy camera traps and provide them with the necessary support. This allows students to learn which animals live around their school and excite them about nature (McShea et al., 2016).

Next to these additional positive effects that result from letting people deploy camera traps themselves or help researchers classify their images, the images from camera trap surveys are being used all over the world to excite people about wildlife. Images are a powerful tool to promote conservation among citizens since they show parts of our world that usually remain unseen (Swann and Perkins, 2014). The images are not only useful to raise global environmental awareness, but also play a role in sensitizing local communities and building conservation management capacity (Rovero et al., 2013).

2.6.5 Privacy considerations

Striving towards an open data policy and sharing data with the general public comes with some important considerations. Nowadays, the protection of citizens' private sphere is a sensitive issue. When images from camera traps mounted in public areas are distributed, there always exists the risk that passers-by have triggered the camera traps. Images

showing people cannot be made public. Before the data can be made available to volunteers, these images need to be removed from the dataset (Steenweg, 2016). This can be automated by, for example, training a neural network to recognise people. Some countries already implemented restrictions on the use of camera traps, for example Austria and Switzerland (Rovero et al., 2013). In Belgium, no specific rules exists yet for the deployment of camera traps, but the general privacy regulations apply also to camera traps (Privacycommissie, 2018). Furthermore, some studies may not want to release geographical locations of endangered or rare species for fear of increasing poaching or to prevent enthusiasts from disturbing the animals and their habitat (Steenweg, 2016).

CHAPTER 3

DESCRIPTION OF THE CAMERA TRAP IMAGES

To train the image classification network to recognize humans and identify animal species, training data is needed. This data consists of camera trap images and their associated labels, indicating the manually annotated content of the images. In this chapter, the structure of the camera trap images and their associated labels is described. At the end of the chapter, an overview is given of the observed bird and mammal species.

3.1 Study area

3.1.1 Hoge Kempen National Park

The images that are used to train the neural network result from an ongoing camera trap survey from the Research Institute for Nature and Forest (INBO) in collaboration with Hasselt University[1]. The camera traps (see Section 2.3) are mounted in the Hoge Kempen National Park, located in the East of the Province of Limburg (Figure 3.1). This nature reserve consists of more than 5700 hectares of pine forest and heathland (Regionaal Landschap Kempen en Maasland, 2015) and is the first and only Belgian National Park (Jeanloz et al., 2016). The camera trap survey is conducted to investigate natural and human factors influencing the behaviour of wild boars (*Sus scrofa*). Wild boar is native to the Eurasian continent, but went virtually extinct in Flanders several decades ago. The species was observed again in 2006 and is expanding rapidly ever since. On the one hand, this can be seen as a conservation success. On the other hand, the return of wild boar gave rise to some challenges, inherent to the fragmented nature in Flanders. Wild boars can have serious ecological, economic and socio-cultural impacts. One of the most topical impacts of wild boar is the damage to agricultural land and gardens. The Hoge Kempen National Park has a limited size and is surrounded by urbanised area, increasing the chance of encounters between wildlife and people (Hasselt University, 2015).

[1] https://www.uhasselt.be/FieldResearchCentre

Figure 3.1: Location of the Hoge Kempen National Park in the East of the Province of Limburg. Image from Jeanloz et al. (2016).

3.1.2 Sampling configuration

During twelve months, 40 camera traps are mounted in the National Park. Every month, the cameras are moved to new locations. A careful sampling design is necessary to contend with spatial variability. This can be addressed by defining a target population of sampling units for inference (for example grid cells) and deploying cameras following a probability-based design (Burton et al., 2015). To determine the location of the camera traps, an imaginary grid with identical cells was placed over the study area, resulting in 640 possible camera locations (Figure 3.2). The National Park was divided into 40 areas of equal size in order to obtain an even distribution of the camera traps over the area. For every month, a set of 40 sampling points was randomly selected, with one location in every zone (Figure 3.3). A camera trap mounted at a certain location is called a deployment. The dataset used to train and evaluate the image classification network consists of the camera trap images taken from May until September 2017. This corresponds to 134 deployments.

3.2 Camera trap images

3.2.1 Image sequences

When the camera is activated, a number of images are taken consecutively. By default the camera takes three pictures per trigger, but this setting can be changed (Reconyx, 2013). During the camera trap survey in the Hoge Kempen National Park, the number of pictures per trigger was set at ten. These images are grouped into a sequence. When the first round of ten images is completed, the camera checks whether there is still movement. If

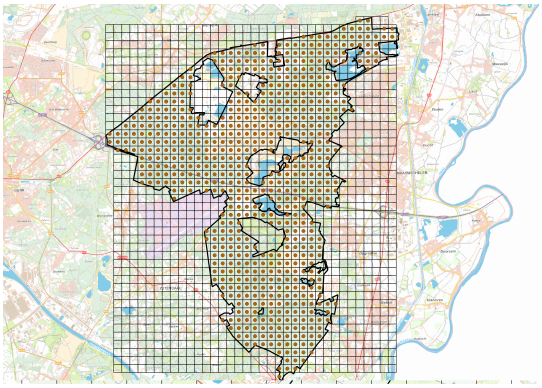


Figure 3.2: Sampling grid to determine the camera locations.

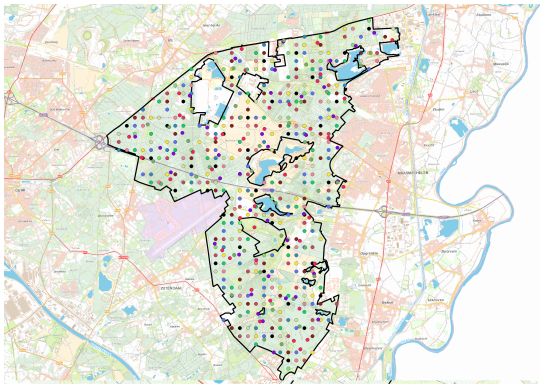


Figure 3.3: Camera trap locations during the first year. Every colour denotes a different set of 40 locations.

so, a second round of ten images is taken. This process will continue until no more motion is detected. Sequences of ten images that are taken immediately after one another, are combined into one large sequence. A sequence therefore consists of at least ten images. The camera switches automatically between night mode and day mode, taking respectively monochrome infrared images and colour images (Reconyx, 2013).

3.2.2 Control images

Normally, the camera only takes images when it is activated by the motion detection sensor. However, twice a day, every twelve hours, one image is taken without the camera being activated by the motion detector. These images can afterwards be used to verify whether the camera was working correctly, since equipment failure is one of the common issues users of camera traps have to deal with (see Section 2.4.2). Otherwise, if there are no images for a long period of time, it would be impossible to know if there were simply no animals passing by or that the camera was malfunctioning. Because of the control images, there is an image available at least every twelve hours. These images are also grouped into a sequence when multiple control images were made consecutively, without any sequences based on motion detection in between them.

3.3 Annotations

The camera trap images that are used to train the neural network are manually annotated using Agouti[2]. This is an application for processing images from camera trap surveys, developed by scientists of Wageningen University in collaboration with INBO. The images are not individually annotated, but labels are assigned per sequence. The animal species,

[2] https://www.agouti.eu

the number of animals and when possible also the sex and age of the animals are entered into Agouti. Additional notes on the animal behaviour can be made. The presence of humans and domestic animals is also indicated. Sequences taken during the set-up and pick-up of the camera and blank sequences are marked.

Since only the sequences are labelled and not the individual animals or even the separate images, only sequences that contain one animal species are used to train the network. Sequences with more than one species are removed from the dataset. This situation is not very common, resulting in seven of the 4316 sequences being removed from the dataset.

3.4 Overview of the data

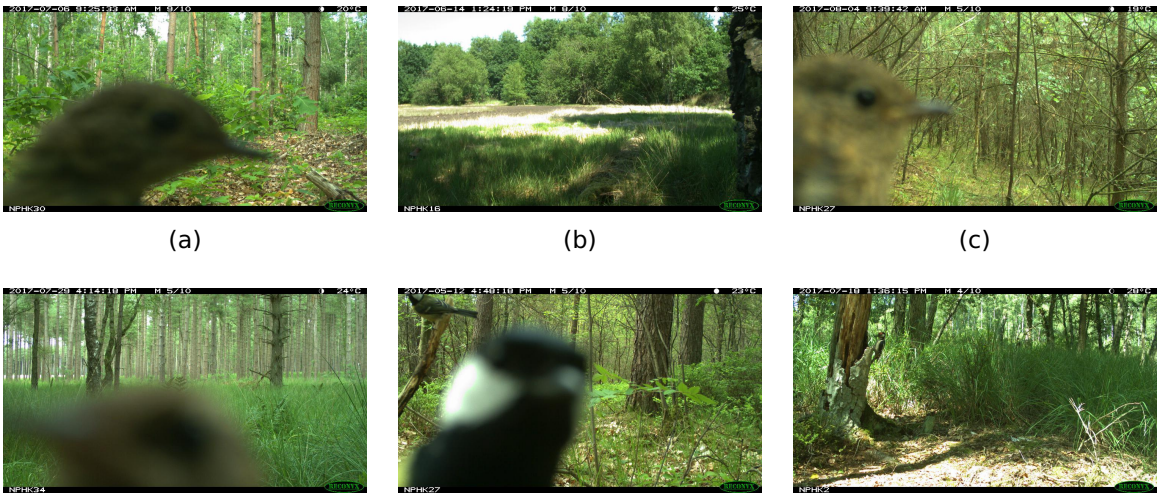
The camera trap images that are used are taken from May until September 2017, at 134 locations (deployments) in the Hoge Kempen National Park. The dataset consists of 4316 sequences or almost 150 000 images. As mentioned above, seven sequences containing more than one species are removed, resulting in 4309 remaining sequences. One sequence containing horses and crows was removed as well as three sequences containing horses and geese and three sequences containing roe deer and wild boars. Table 3.1 gives an overview of the available sequences and observed species. Thirteen mammal species and ten bird species were observed. The number of sequences per annotation is highly variable, resulting in an unbalanced dataset (see Section 4.3).

3.4.1 Bird species

As can be seen in Table 3.1, most of the observed bird species only appear in a small number of sequences. As determined by Swann et al. (2004), animal size and speed influences trapping rates. Furthermore, in contrast to sequences containing large mammals, in most of the bird sequences, only the first image(s) of the sequence contain(s) the bird, resulting in even less images of birds. These images are also often not ideal images. Only part of the bird is in the image or the bird is very close to the camera resulting in images being out of focus or the opposite, the bird is far way from the camera. Figure 3.4 displays the camera trap image of six bird species that most clearly shows the animal, illustrating this problem. All of this makes it difficult to identify the bird species. Therefore, all ten observed bird species are combined into one class.

Table 3.1: Overview of the number of sequences per annotation. (Sequences containing more than one animal species are discarded.)

Vernacular name	**Scientific name**	
Ass	*Equus asinus*	8
Beech Marten	*Martes foina*	17
Carrion Crow	*Corvus corone*	3
Common Pheasant	*Phasianus colchicus*	1
Domestic Cat	*Felis catus*	14
Domestic Dog	*Canis familiaris*	4
Eurasian Blackbird	*Turdus merula*	24
Eurasian Jay	*Garrulus glandarius*	1
Eurasian Red Squirrel	*Sciurus vulgaris*	6
European Hare	*Lepus europaeus*	10
Great Spotted Woodpecker	*Dendrocopos major*	1
Great Tit	*Parus major*	9
Greylag Goose	*Anser anser*	15
Horse	*Equus caballus*	30
House Sparrow	*Passer domesticus*	10
Long-tailed Field Mouse	*Apodemus sylvaticus*	8
Red Fox	*Vulpes vulpes*	236
Sheep	*Ovis aries*	9
Short-toed Treecreeper	*Certhia brachydactyla*	1
Song Thrush	*Turdus philomelos*	1
Western European Hedgehog	*Erinaceus europaeus*	2
Western Roe Deer	*Capreolus capreolus*	1282
Wild Boar	*Sus scrofa*	190
Human		27
Blank		2092
PickupSetup		308
Total		**4309**



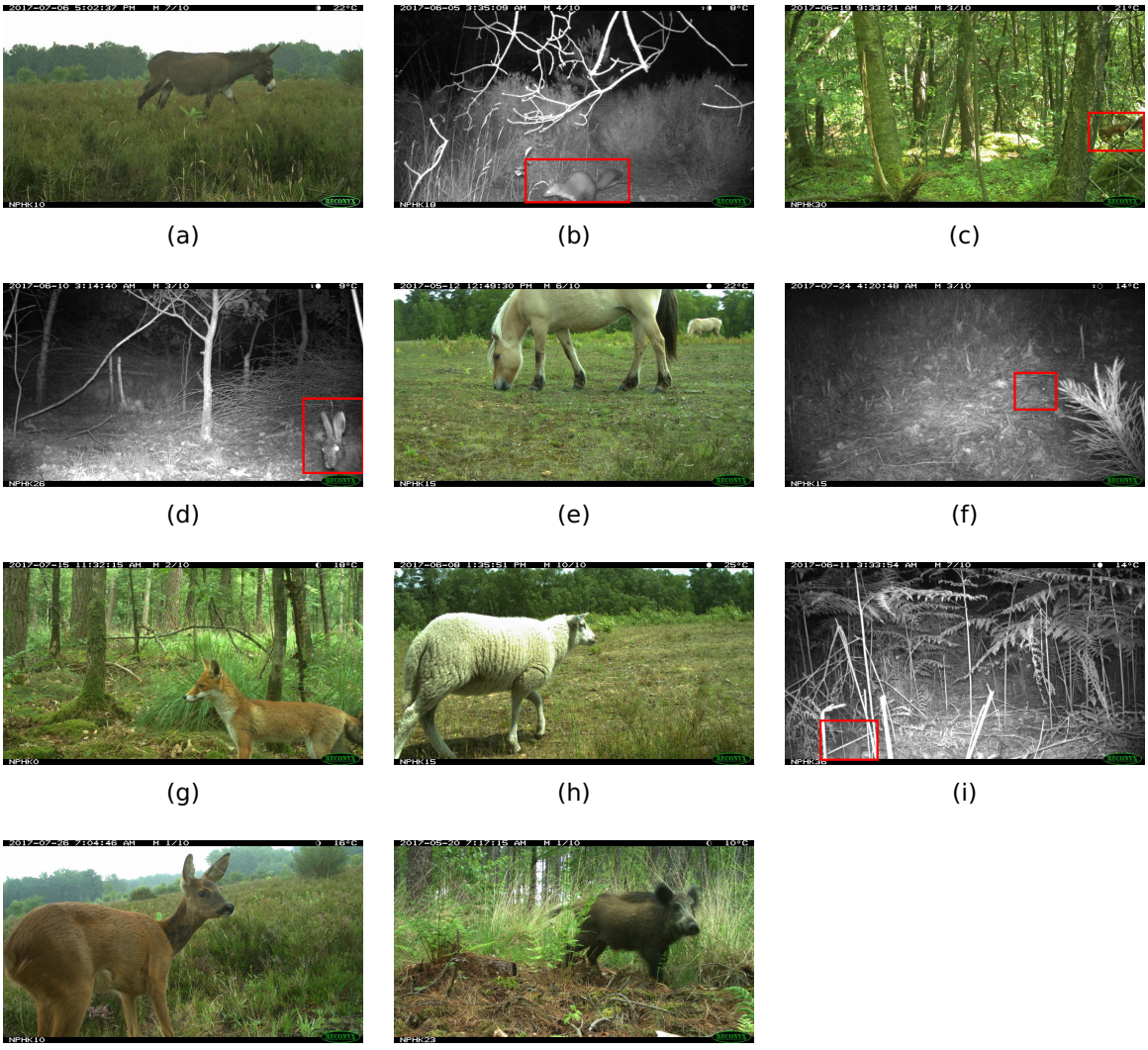
(a) (b) (c)

(d) (e) (f)

Figure 3.4: Camera trap image available in the dataset that most clearly shows (a) the Eurasian blackbird, (b) the Eurasian jay, (c) the house sparrow, (d) the short-toed treecreeper, (e) the great tit and (f) the great spotted woodpecker. The great spotted woodpecker is sitting on the tree on the left side of the image. The Eurasian jay is hidden in the grass in the bottom left corner of the image and is nearly impossible to find, even when the image is enlarged.

3.4.2 Mammal species

Figure 3.5 shows camera trap images of all observed mammal species, except the domestic cat and dog. These images were chosen to have a clear representation of the animal species. In reality, not all images are perfect, as explained in Section 4.2. Therefore, for some species even the most ideal image does not show the animal clearly. If needed, rectangles were added to indicate the position of the animal.



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

(j) (k)

Figure 3.5: All observed mammal species, except the domestic cat and dog: (a) ass, (b) beech marten, (c) Eurasian red squirrel, (d) European hare, (e) horse, (f) long-tailed field mouse, (g) red fox, (h) sheep, (i) Western European hedgehog, (j) Western roe deer and (k) wild boar. If needed, rectangles were added to indicate the position of the animal.

CHAPTER 4

IMAGE CLASSIFICATION

To automatically recognise people and identify animal species, an image classification model needs to be built. This model will assign a label to the camera trap images from a fixed set of possible labels. When classifying camera trap images, there are some additional challenges due to the environmental conditions in which the camera traps are mounted, the behaviour of the animals and hardware limitations of the camera traps. Additionally, the dataset is heavily imbalanced, meaning that some classes are much more common than others. Finally, since the training data consist of annotated sequences, not individually labelled images, a suitable method needs to be selected to aggregate the predictions for the individual images into one label for the whole sequence.

4.1 Image classification models

An image classification problem is the task of assigning a label from a fixed set of possible labels to an input image. An image is actually just a large array of numbers which has to be converted to one single label. The model assigns probabilities to the possible labels after which the most likely label can be determined (Karpathy et al., 2016). In the case of identifying animals in camera trap images, the possible labels are the different species observed in the area. Note that when a new animal species occurs that has not been observed before, this species will not be included in the labels available to the model and therefore will not be classified correctly.

It would be an enormous task trying to specify the characteristics of each class so that an object can be identified based on a set of rules. Therefore, the model is trained using many manually labelled images of each class and in this way learns the visual appearance of each class. This approach is referred to as a data-driven approach (Karpathy et al., 2016).

4.2 Challenges when classifying camera trap images

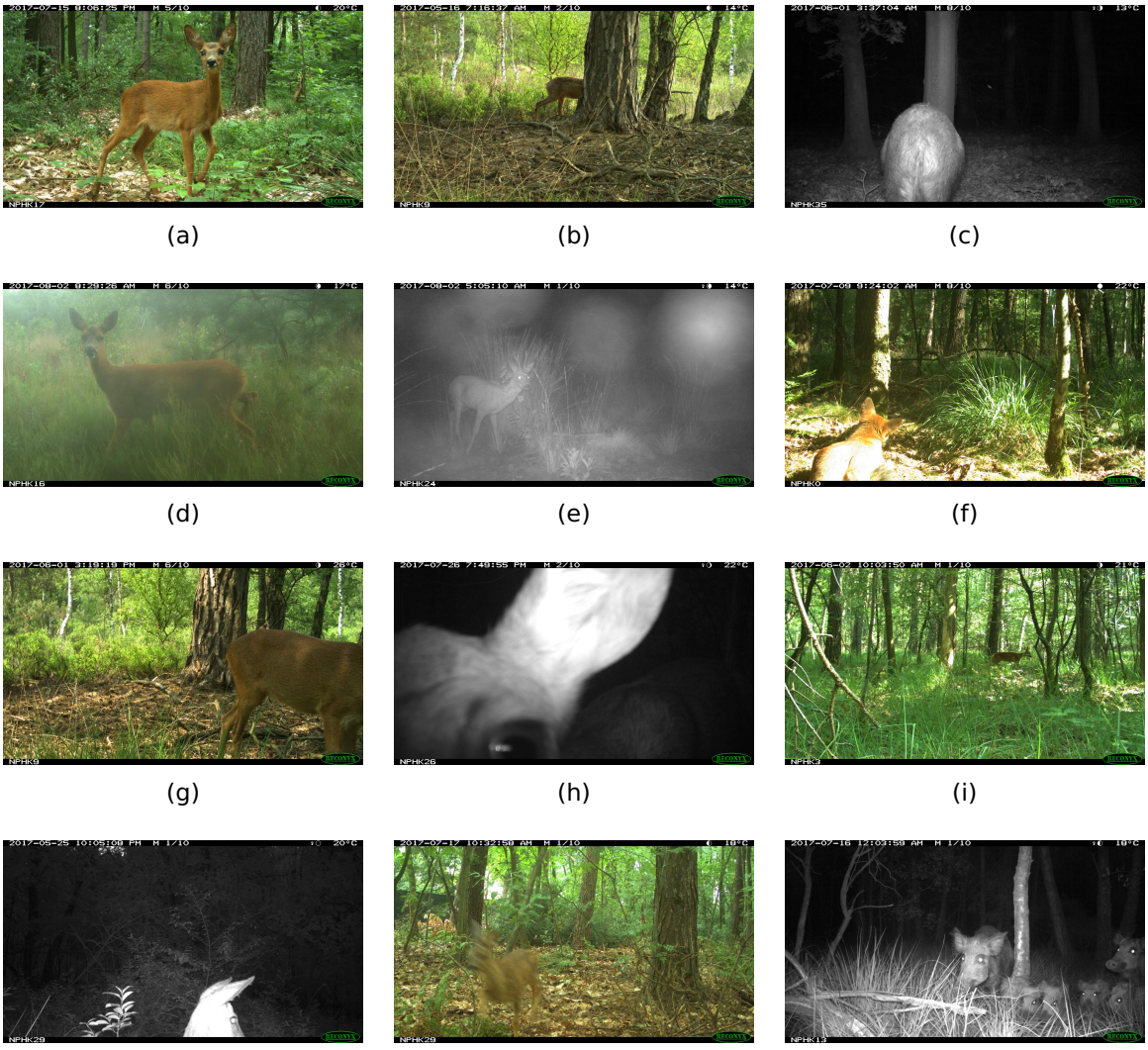
The task of visual recognition is trivial for humans, but comes with some challenges for computer algorithms (Karpathy et al., 2016). When working with camera trap images, the

image conditions become even more challenging (Yu et al., 2013). This results in automatic classification of animal species in camera trap images still remaining a challenging problem (Gomez et al., 2017). For camera trap images, the challenges can be subdivided in three main groups: environmental conditions, animal behaviour related and hardware limitations (Gomez et al., 2017). Figure 4.1 shows some examples of these challenges.

Environmental conditions refer to how the rough conditions the camera is mounted in, affect the quality of the image. Since camera traps are deployed in a natural environment with vegetation, many objects can occlude the target animals (Gomez et al., 2017). This environment is not static (see Section 5.3), so even if there was no occlusion when placing the cameras, it can appear at any moment, for example when branches are moved into the field of view of the camera. Another challenge is the changing illumination conditions whereby the colour of an object, the pixel values of the array, changes during the day and between different days (Karpathy et al., 2016). Day and night have very different illumination conditions but especially the transition between them causes problems (Gomez et al., 2017). Figure 4.1f shows an example of overexposed regions caused by the sunlight. Varying weather conditions such as rain are also examples of conditions that directly affect hardware performance (Gomez et al., 2017). Figure 4.1e shows an image taken with raindrops on the lens.

A vast majority of camera trap images do not contain the whole animal due to the above-mentioned context occlusion or when the animal is too close to the camera or when a part of the animal is simply outside the field of view of the camera (Karpathy et al., 2016). Depending on the position of the animal, a picture can be taken from many different angles. Another animal behaviour related challenge is that animals can move their bodies and therefore do not always have the same shape and appearance (Karpathy et al., 2016). A running roe deer for example will look different compared to his congener who is lying down on the ground. Furthermore, important features to recognize a species can be hidden because of the complex pose of the animal. Particularly for similar species this can be a problem (Gomez et al., 2017). In addition, the size of animals of the same species can vary in the image (Karpathy et al., 2016). An animal can have a larger extent in the image because it is closer to the camera, but also the real size can vary between individuals of the same species. Furthermore, an image may include multiple animals, even different species. As explained in Section 3.3, sequences with more than one species were removed from the dataset.

Challenges related to camera trapping hardware and selected parameters are blurred images and overexposed animals due to the infrared flash. This results in shape and fur patterns being indistinguishable, increasing the difficulty of the classification task (Gomez et al., 2017).



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

(j) (k) (l)

Figure 4.1: Different camera trapping classification scenarios: (a) ideal, (b) occlusion due to context, (c) auto-occlusion, (d) disturbing weather conditions, (e) raindrops on the lens, (f) poor illumination conditions, (g) part of the animal, (h) animal close to the camera, (i) animal far away from the camera, (j) overexposed animal, (k) blurred animal and (l) multiple animals and different sizes.

An automatic species recognition system must be able to deal with all these challenging conditions and classify an image only with partial information. Because of all the different situations in which an animal can appear, the classification model must be invariant to all these variations, but at the same time be sensitive to the variations between the different classes (Karpathy et al., 2016).

4.3 The class imbalance problem

As mentioned in Section 3.4, the dataset is heavily imbalanced. There are big differences between the number of sequences that are available to train the neural network to recognise the different classes. This results in a significant difference between the prior probabilities of the different classes. This situation is known as the class imbalance problem (López et al., 2013). The problem of an imbalanced dataset is that models can limit their predictions to the most frequent classes and still achieve a high level of accuracy. Often it is in particular the smaller classes that one is interested in (Norouzzadeh et al., 2017). For example, in the research project the training images result from, one is interested in wild boar (*Sus scorfa*), appearing in only 190 of the 4309 sequences (see Table 3.1). Many solutions have been proposed to deal with the class imbalance problem. Two commonly used types of methods are data sampling and cost-sensitive learning (López et al., 2013).

4.3.1 Data sampling

The general principle of the data sampling approach is to preprocess the training data and modify it in such a way that a more or less balanced set is obtained (López et al., 2013). There are three types of resampling methods: undersampling, oversampling and a hybrid method that combines under- and oversampling. When performing undersampling, a subset of the original dataset is created by eliminating instances of the majority class. However, the imbalanced class problem is generally associated with binary classification, but the multi-class problem often occurs as well, for example when classifying camera trap images. In this setting, there can be several minority classes, making it more difficult to solve the imbalance problem, especially when using resampling methods (López et al., 2013). The opposite of undersampling is oversampling, where a superset of the original dataset is created by replicating some instances or by creating new ones based on the existing ones. The major drawback of resampling methods is that potentially useful data is discarded when performing undersampling or that the likelihood of overfitting is increased when using oversampling (López et al., 2013).

4.3.2 Cost-sensitive learning

This method incorporates solutions at both the data level and the algorithmic level. Cost-sensitive learning takes into account the variable cost of a misclassification with respect to the different classes. For classification tasks, the measure of performance is often defined as the proportion of examples that the models correctly classifies (Norouzzadeh et al., 2017). To conquer the imbalancedness of the dataset and avoid that the model achieves a high performance by only learning the majority classes, higher costs are assigned to misclassification of examples from the minority classes (López et al., 2013). The challenge when applying this technique is that a suitable cost matrix must be determined. These cost values can be given by domain experts or can be derived from the data.

Norouzzadeh et al. (2017) propose a weighted loss approach for the classification of camera trap images with imbalanced classes. A higher cost is put on misclassifying images from rare classes and a lower cost on misclassifying examples of the frequent classes. The importance f_i of each class is computed as follows:

$$f_i = \frac{N}{n_i}, \tag{4.1}$$

with N the total number of images in the set and n_i the total number of images for each class i in the training set. The weights w_i for each class are then calculated using the following formula:

$$w_i = \frac{f_i}{\sum_{i=1}^{k} f_i}, \tag{4.2}$$

with k the number of classes, so that the sum of the weights of all classes is equal to one. When datasets are highly imbalanced, this approach could cause extremely high and low weights, resulting in very small or very large gradients, which can be harmful to the learning process. To avoid this, the gradients can be limited within a certain range (Norouzzadeh et al., 2017). Section 6.4 explains how a neural network is trained, including the role of the gradient in this process. The research of Norouzzadeh et al. (2017) shows that this method to conquer the imbalancedness of the data can increase the accuracy for the rare classes, while keeping the same level of accuracy for most of the other classes.

4.4 Classifying image sequences

The dataset contains labels for sequences, not individual images. However, the neural network will be trained to classify images. The label of the sequence is assigned to every image of the sequence. This introduces additional noise into the dataset since the animal

is not always in every image of the sequence. Afterwards, the individual predictions need to be aggregated to predict one label for the entire sequence. It should be possible to train a neural network to directly classify sequences, but this causes challenges related to sequences with different numbers of images and larger neural network sizes, amongst other things (Norouzzadeh et al., 2017).

A possible way to aggregate the individual predictions is to simply select the most frequently predicted label. An other possible approach is averaging the top-5 predictions of all the images of the sequence, as described by Norouzzadeh et al. (2017). For each image, the model outputs a probability distribution over all classes from which the top-5 guesses can be selected. Across all images of the sequence, the probabilities are summed up by class and divided by the total number of images N. The result is a vector of probabilities for different classes, with a length varying from 5 to $5N$. The final aggregate prediction is the label with the highest probability. Norouzzadeh et al. (2017) found that the accuracy scores for sequences consisting of two to three images are on average 0.5% higher than those for individual images.

CHAPTER 5

IMAGE PREPROCESSING

In order to improve the performance of the neural network, it is helpful to select the regions of interest of the camera trap images and feed only those to the input layer of the neural network. In the case of camera trap images, the regions of interest of the image can be an animal, a human or even a car in the background of the image when the camera is positioned close to a road. The neural network architecture that will be used, requires fixed input dimensions. Therefore, all extracted regions must have the same size. However, in one camera trap image multiple areas can be selected, depending on the number of interesting objects in the image and their size. In order to achieve this, the images are preprocessed. The preprocessing is shown schematically in Figure 5.1 and is step by step described in more detail in this chapter. The result of the different steps is illustrated by means of an example sequence (Figure 5.2). This sequence consists of ten images and shows a wild boar running across the field of view of the camera, from left to right.

5.1 Benefit of using segmented images

Gomez et al. (2017) compared the performance on a dataset consisting of manually segmented images to the performance on the dataset containing the original camera trap images, for eight different neural network architectures. For all network architectures, the performance was higher when the segmented images were used. Another problem that might occur when using the whole image as an input for the classifier, is that the predictions are influenced by the background of the image. For example, Ribeiro et al. (2016) found that their classifier could be misled by the presence of snow in the image when discriminating between wolfs and huskies. Shane (2018) gives an example of a neural network predicting the presence of sheep, just because the image shows hilly green fields, without sheep being present. This is because of the way neural networks learn (see Section 4.1). When a lot of the example images show sheep in hilly green fields, it is possible that the classifier associates the label ‘sheep’ with the background, the green fields, and not with the object of interest, the animal. So when a sheep shows up in a for the network unexpected place, it has a hard time classifying it (Shane, 2018).

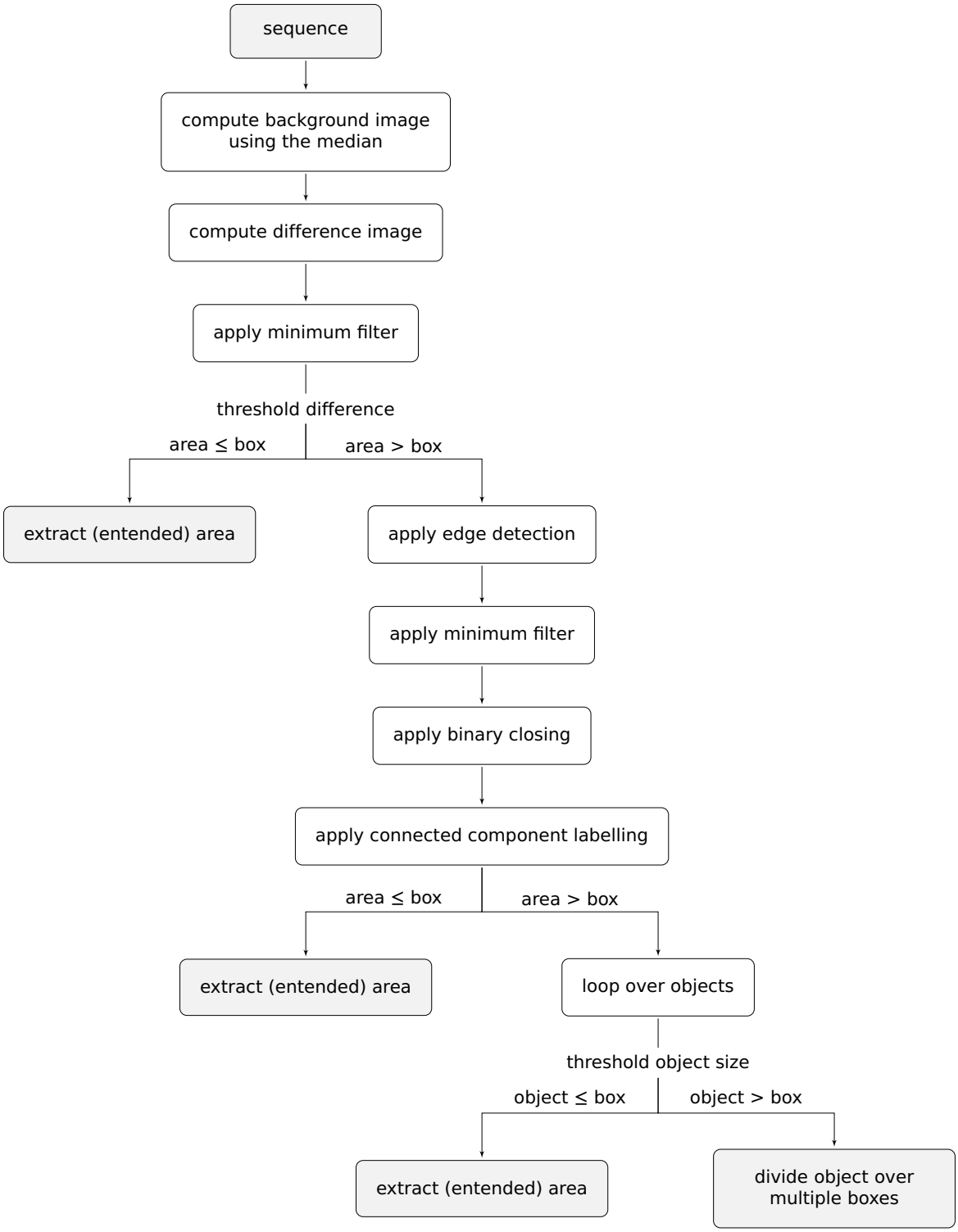


Figure 5.1: Preprocessing flowchart used to find the regions of interest in an image.

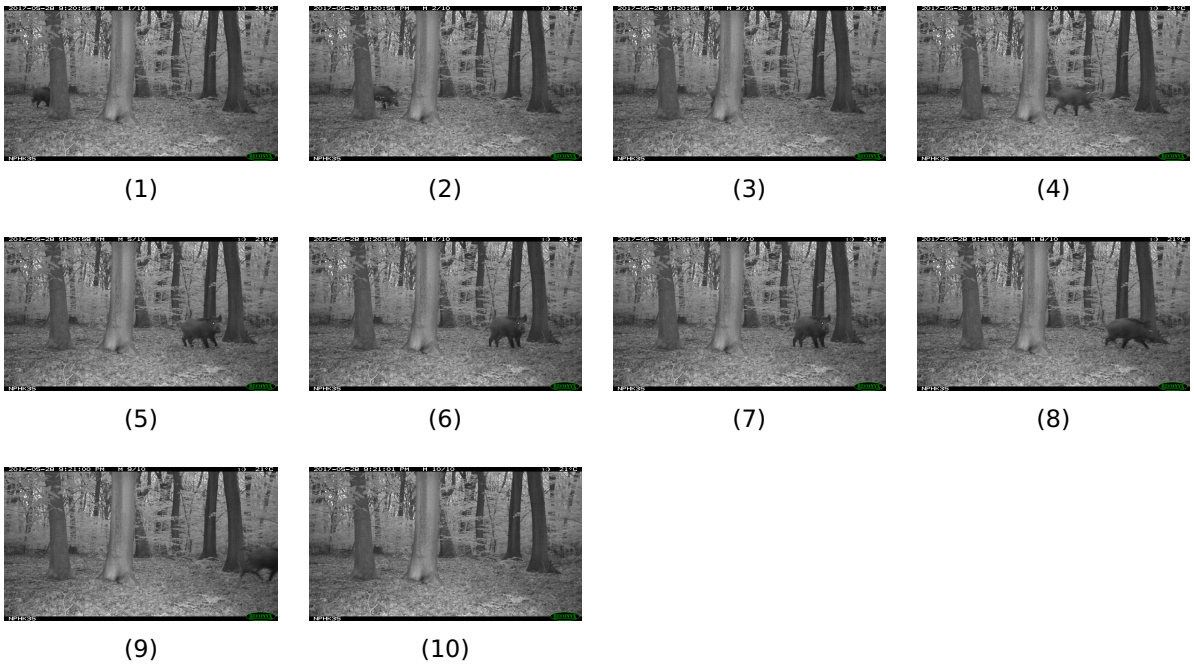


Figure 5.2: Example sequence to illustrate the result of the different steps of the preprocessing.

An additional advantage of determining the regions of interest in the camera trap images, not related to the performance of a neural network, is that a box can be drawn on the camera trap image, indicating those region of interest. When images are manually processed, finding the animal or deciding that there is no animal in the image is typically the most time consuming part (McShea et al., 2016). Automatically drawing boxes around the regions of interest may ease this process.

5.2 Size reduction

The original camera trap images have a resolution of 1920 by 1080. To reduce the computational load of the preprocessing and when training the neural network, the dimensions of the images are reduced 50%, resulting in images with a resolution of 960 by 540. This size reduction might sound dramatic, but is barely noticeable, as shown in Figure 5.3. The preprocessing was first optimized based on the original images. Afterwards, it was explored how much the resolution of the images could be reduced without having an adverse effect on the performance of the segmentation. The minimum and maximum pixel difference (see Section 5.4.1) were changed in proportion to the size reduction, as well as the size of the extracted area. The value of other parameters used during the preprocessing could remain the same.

All of this was determined based on visual assessment. The regions of interest were not extracted from the images, but instead indicated on the image, making it possible to evaluate

(a) Original (b) Resized

Figure 5.3: Original camera trap image (1920x1080) (a) and resized image (960x540) (b).

their position. A test set was composed containing 30 sequences with different situations to determine the optimal operations and their parameters. This test set consists of sequences with one animal, sequences with multiple animals, sequences with small animals, sequences with large animals, empty sequences (false triggers), sequences with humans, sequences taken during the camera set-up, et cetera. In addition, for a final visual assessment, the resulting boxes of five deployments were plotted on the images and evaluated.

5.3 Background image

To select the regions of interest in an image, a standard, background image is needed to be able to determine which elements in the image are different compared to the normal situation.

A possible approach is using the control images to compare the images to. These control images are automatically taken every twelve hours without the camera being triggered to be able to check if the camera is still working. These images thus represent the background situation. However, during the day, the light changes and therefore also the shadow patterns. This problem was also reported by Swinnen et al. (2014) who tried to discriminate between recordings of target species and non-target recordings based on detecting variation (changes in pixel values) in the recordings. Furthermore also branches and leaves close to the camera can be moved substantially by wind or by animals passing by. The background is therefore not static and with the cameras being installed in a natural environment, the background situation may change significantly during the presence of the camera trap and even during the small time window between an image and the closest control image. These changes were too big to be able to correct them, so the control images were not usable to construct a background image.

An advantage of the way the camera works is that it takes at least ten pictures when it is triggered. When it is still triggered after these ten images, it takes ten more images and so on. Every image is thus part of a sequence of at least ten images. This sequence can

(a) Median (b) Average

Figure 5.4: Background situation computed using the median (a) and the average (b). On the average image (b) two horizontal red line are added to indicate the position of the remaining haze.

be used to construct a background image. Since the sequence is taken during a very short time period, the light changes and other changes to the background are minor. Weinstein (2015) uses a running average to represent the background, with a parameter controlling the degree to which more recent frames contribute to the new background state. However, Weinstein (2015) works with time-lapse videos and has a lot more frames available per time unit than supplied by our camera trap. Also, since our camera only takes images when it is triggered, except for the control images, the changes between sequences can be big due to the different causes mentioned above. So instead of using a running average over all the images of a deployment, the computation of the background image is restricted to the images of the sequence and a new background image is constructed for every sequence. In addition, since the smallest sequences only exist of ten images, a more robust measure (Huber, 2011), the median, is used to determine the background situation instead of the average. Figure 5.4 illustrates the difference between the use of the median and the average to compute the background for a sequence consisting of ten images. If one looks closely, one can see that on the average image a haze of the roe deer running across the image from left to right remains (in between the two red horizontal lines), while on the median image the background situation is visible without any haze. The median value is a pixel value that actually occurs in one of the images of the sequence, while the average value is computed based on the pixel values of the sequence and therefore not necessary occurs in one of the images of the sequence, resulting in a haze.

However, the very first step in the preprocessing of the images, before the median images and differences can be calculated, is the removal of the borders. At the top and bottom of the image, a black border is present. On those borders the metadata of the image, such as the time and temperature, is stamped. These borders can simply be removed by cropping the image. In all following steps, the cropped images are used. Figure 5.5 shows the computed background image using the median for the example sequence.

Figure 5.5: Computed background image for the example sequence (Figure 5.2) using the median.

5.4 Filtering

Before filtering, the difference images of the colour images are converted to greyscale images, since for the remaining part of the preprocessing only the presence of differences and their gradients are important (see Section 5.5.1). All further preprocessing is done using greyscale images with only one channel, either original infrared greyscale images or converted colour images. At the end of the preprocessing, when the location of the regions of interest of the image is determined based on the greyscale images, the regions are extracted from the original images.

The difference between an image and the median image of the sequence, the background state, is only a first rough estimate of the regions of interest of the images. Not only the objects of interest are different in comparison to the background image, also other elements may have moved slightly during this small time period. Noise removal is therefore necessary. A challenge when doing the noise removal is that the method has to be coarse enough to remove for example moving leaves, which can occupy a large area on the image when they are close to camera, but not too rough so that small objects of interest, for example young wild boars, are not removed. Figure 5.6a shows the original, unfiltered difference between the sixth image of the example sequence (Figure 5.2) and the median image (Figure 5.5) and Figure 5.6b shows the difference after filtering.

5.4.1 Minimum filter

As a fist step to remove noise, a minimum filter is passed over the difference image. Based on visual assessment of the result (see Section 5.2), a filter with a kernel size of nine was selected. This means that the value of a pixel is changed to the lowest pixel value in the nine by nine neighbourhood of this pixel.

When the difference between an image and the median background image is computed, the absolute value of the difference is taken. The lowest value in the difference image is therefore zero, indicating that there is no difference compared to the background state. Pixel values higher than zero indicate a change compared to the background. When the

(a) Unfiltered difference (b) Filtered difference

Figure 5.6: Difference between the sixth image of the example sequence (Figure 5.2) and the median image (Figure 5.5) before and after filtering.

minimum filter is passed over the difference image, the value of all pixels is changed to the lowest pixel value in their neighbourhood. When the neighbourhood of a pixel contains a zero difference pixel, the value of that pixel is set to zero, removing some of the noise. The minimum filter unfortunately not only removes noise, but also removes part of the object if a zero difference pixels is in its neighbourhood. A kernel size of nine provides a good balance between noise removal and not removing too many of the object pixels, especially in the case of small objects.

After filtering, a minimum and maximum threshold is applied to the number of pixels that are different compared to the background image. Images with too little difference are discarded. When the number of pixels is too small, the difference is due to remaining noise and not due to an object of interest. Differences larger than the upper threshold are mostly images that are taken during the set-up or pick-up of the camera, whereby the camera shakes and the field of view of the camera changes abruptly. This results in a useless background image when computing the mean values and therefore (almost) every pixel will be different, so no objects can be detected. Another possibility is that an animal appears very close to the camera, occupying the whole image. Images with too much difference are not processed further, but the whole image is divided into areas with a size equal to the predefined size of the input images of the neural network.

5.5 Image segmentation

As mentioned before, all extracted areas must have the same predefined size. After the first filtering step, the size of a rectangular area containing all pixels having a value larger than zero is determined. If the dimensions of that area are smaller than the predefined size, the extraction area is extended to fit the predefined size. If the dimensions are larger, the area has to be subdivided into smaller parts. To do so, we will try to detect the different objects in the difference image. Small objects, for example several young animals, might

Figure 5.7: Result after edge detection and subsequent filtering for the sixth image of the example sequence (Figure 5.2).

Figure 5.8: Result after binary closing for the sixth image of the example sequence (Figure 5.2).

be grouped together in one box, while larger objects, for example an animal close to the camera, might have to be divided over more than one box.

5.5.1 Edge detection

To determine the position of the object, edge detection is used. Edge detection identifies sharp discontinuities in an image. These discontinuities can be either changes in the image intensity or in the first derivative of the image intensity (Senthilkumaran and Rajesh, 2009).

One of the most frequently used edge detection methods, Roberts Edge Detection (Senthilkumaran and Rajesh, 2009), led to the best result, again based on visual assessment (see Section 5.2). Roberts Edge Detection, also called the Roberts Cross operator, is based on the approximation of the two-dimensional spatial gradient on an image by computing the sum of the squares of the differences between diagonally adjacent pixels. The input is a greyscale image and so is the output. The pixel values of the output represent the estimated absolute magnitude of the spatial gradient at every point (Senthilkumaran and Rajesh, 2009).

A disadvantage of the used edge detection method, is that it enhances noise. Therefore, the output image is filtered again. This time a minimum filter with a kernel size of three is used and passed twice over the image to remove noise. Figure 5.7 shows the result of the preprocessing after edge detection and subsequent filtering for the sixth image of the example sequence (Figure 5.2).

5.5.2 Binary closing

After edge detection, only the edges of the objects are highlighted. The result is not a smooth edge surrounding every object, but multiple separate pieces of edge. These pieces need to be connected to each other to construct the different objects in the image. This is done using binary closing.

Before applying binary closing to the result of edge detection and filtering, the greyscale image is converted into a binary image. Pixel values of zero represent the background and pixel values of one indicate the foreground.

Binary closing combines erosion and the opposite process of erosion, dilation. When an erosion filter is applied to an image, the boundaries of foreground objects are eroded away. A foreground pixel will only remain one if all the pixels in its neighbourhood are also one, otherwise it is set to zero. The opposite process is dilation where a pixel value is set to one if at least one pixel in its neighbourhood is one. It increases the size of an object (Haralick et al., 1987). The neighbourhood of a pixel is defined by a structuring element, a matrix of zeros and ones. Binary closing is in fact the erosion of the dilation of the image with the same structuring element. Closing therefore fills holes smaller than the structuring element (Haralick et al., 1987). The structuring element used here is a 20 by 20 fully-connected kernel. The dilation step of the closing and then the erosion step are each repeated five times. Figure 5.8 shows the result of applying binary closing for the sixth image of the example sequence (Figure 5.2.)

5.5.3 Connected component labelling

The last step in selecting the regions of interest of the image and dividing them over boxes of a predefined size, is to identify the different objects in the processed difference image. This is done via connected component labelling. Two pixels are connected when they are neighbours and have the same value (Haralick and Shapiro, 1985). All eight neighbours of a pixel are considered. Remember that the result of binary closing, as the name suggests, is a binary image. All foreground pixels and all background pixels have the same value, respectively one and zero. The result of applying connected component labelling to the output image of binary closing is an image in which all neighbouring foreground pixels are grouped together, getting the same label. Since there is only one object in the example sequence (Figure 5.2), all foreground pixels get the same value, resulting in the same image as in Figure 5.8. When there are multiple objects, they each get a different label, represented by their pixel values, resulting in the objects having different colours.

Before the objects can be extracted, a threshold is applied to the size of the objects to remove any remaining noise. All objects that consist of a number of pixels smaller than the threshold are discarded. If all objects fit in one box, only one rectangular part of the image is extracted. When this is not the case, the individual objects are extracted. If a single object is still too large, the object is divided over multiple boxes. Figure 5.9 shows the area that will be extracted from the sixth image of the example sequence (Figure 5.2) and the extracted part that will be passed to the neural network.

(a) (b)

Figure 5.9: Area that will be extracted from the sixth image of the example sequence (Figure 5.2) (a) and the extracted part that will be passed to the neural network (b).

5.6 Data after preprocessing

Table 5.1 gives an overview of the available sequences and observed species after preprocessing the images. The last column of this table indicates the difference between the number of sequences before and after preprocessing. These differences can be explained by the minimum threshold that is applied to the number of pixels that is different compared to the background situation. Images with too little difference are discarded, as explained in Section 5.4.1. When all images of a sequence show too little difference, the whole sequence disappears. For the 'Blank' sequences, this is the intention. However, as explained in Section 5.4, there is a trade-off between noise removal and preserving objects of interest that occupy only a small area of the image, for example small animals, parts of animals or animals far away from the camera. Therefore, not all empty sequences are removed during the preprocessing. For example, sequences resulting from the camera being triggered by moving branches show a sufficient amount of difference compared to the background image. Therefore, the neural network will be trained to recognise empty camera trap images. Also some sequences taken during the set-up or pick-up of the camera consist of empty images or are not very different from the background situation. On the other hand, also as a result of this trade-off, some objects of interest that occupy only a small area of the image are lost during the preprocessing. This happened to some sequences containing birds, including the only sequence containing an Eurasian jay. An image of this sequence is shown in Figure 3.4b. As explained in Section 3.4.1, the images of birds are often not ideal and birds are frequently very small in the pictures, resulting in birds being overlooked during the preprocessing. The same applies to some sequences of very small mammals, for example mice.

Table 5.1: Overview of the number of sequences per annotation after preprocessing, divided between day time colour images and night time infrared images. The second last column shows the number of sequences before preprocessing (Table 3.1) and the last column indicates how many sequences were lost during preprocessing.

	After preprocessing			**Original dataset**	**Difference**
Annotation	**Day**	**Night**	**Total**		
Ass	4	3	7	8	1
Beech Marten	0	17	17	17	0
Carrion Crow	3	0	3	3	0
Common Pheasant	1	0	1	1	0
Domestic Cat	2	12	14	14	0
Domestic Dog	4	0	4	4	0
Eurasian Blackbird	22	1	23	24	1
Eurasian Jay	0	0	0	1	1
Eurasian Red Squirrel	5	1	6	6	0
European Hare	4	5	9	10	1
Great Spotted Woodpecker	1	0	1	1	0
Great Tit	9	0	9	9	0
Greylag Goose	14	0	14	15	1
Horse	23	5	28	30	2
House Sparrow	5	4	9	10	1
Long-tailed Field Mouse	0	5	5	8	3
Mouflon	9	0	9	9	0
Red Fox	24	203	227	236	9
Short-toed Treecreeper	1	0	1	1	0
Song Thrush	1	0	1	1	0
Western European Hedgehog	0	2	2	2	0
Western Roe Deer	580	685	1265	1282	17
Wild Boar	20	164	184	190	6
Human	18	8	26	27	1
Blank	1385	199	1584	2092	508
PickupSetup	215	5	220	308	88
Total	2350	1319	3669	4309	640

CHAPTER 6

CONVOLUTIONAL NEURAL NETWORKS FOR IMAGE CLASSIFICATION

Convolutional neural networks will be used to recognise people and identify animal species. Nowadays, convolutional neural networks are the go-to technique for computer vision problems (Chollet, 2018). Although the area of neural networks was originally inspired by the desire to model biological neural systems, it has diverged and become a successful technique for machine learning tasks (Karpathy et al., 2016). This chapter start with a brief introduction to machine learning and deep learning, whereafter the architecture and training of neural networks is explained. Convolutional neural networks are highlighted, since this is the type of neural network that will be used to classify the camera trap images. The network will not be build and trained from scratch, but a pretrained neural network will be adjusted to hierarchically classify the camera trap images.

6.1 Deep learning with neural networks

Deep learning is a specific subfield of machine learning. Machine learning applies a new way of programming: instead of manually implementing data processing rules, the model distills these rules automatically by looking at the data. A machine learning model is trained, rather than explicitly programmed (Chollet, 2018). As explained in Section 4.1, the model is presented with many labelled examples from which it learns the visual appearance of each class. These learnt rules can then be applied to new images to predict their labels. This stands in contrast to classical programming, where a human inputs rules and data is processed according to these rules. Figure 6.1 visualises the difference between classical programming and machine learning.

In deep learning, emphasis is placed on learning successive layers of increasingly meaningful representations. This is usually done via neural networks, consisting of stacked layers.

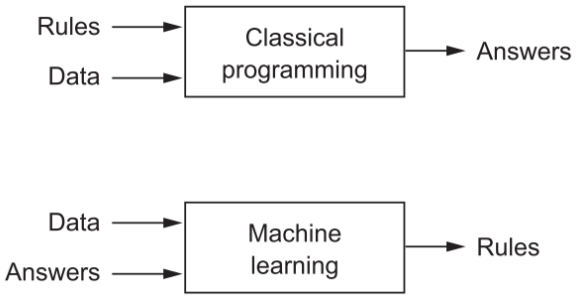


Figure 6.1: Difference in approach between classical programming and machine learning. Image from Chollet (2018).

A deep network can be seen as a multistage information distillation operation. The inputs are mapped to the targets via a sequence of simple data transformations in the layers. The specifications of these transformations are stored in the layer's weights. Training the network means finding the appropriate weights for all layers so that the inputs are correctly mapped to their associated labels (Chollet, 2018). Applications of deep learning include near-human-level image classification, speech recognition and handwriting transcription. Deep learning in also used to improve text-to-speech conversion, for autonomous driving and in digital assistants such as Google Now and Amazon Alexa (Chollet, 2018).

6.2 Architecture of neural networks

Neural networks consist of several types of layers. These layers are the core building blocks of the network and process the data. A layer can be seen as a filter for the data and extracts representations out of this data. Neural networks are chains of mathematical operations, layer by layer, which are just geometric transformations of the input data (Chollet, 2018). The first layer of a network, receiving the input, is referred to as the bottom of the network, while the last layer of the network, providing the output, is referred to as the top of the network. Chollet (2018) presents an easily understandable analogy for a neural network being a complex geometric transformation in a high dimensional space, implemented via a sequence of simple steps. Imagine two sheets of paper crumpled together into a small ball. That ball is the input data and each sheet represents a class in a classification problem. The neural network will need to figure out a transformation of the paper ball to uncrumple it, so that the two classes are easily separable. With deep learning, this is implemented as a sequence of simple transpirations of the three-dimensional space, similar to what one would do when uncrumpling a paper ball, one movement at a time.

Each layer of a network consists of individual neurons. These neurons have learnable weights and biases. Every neuron receives input signals x_i, computes a dot product with the weights w_i, adds a bias b and applies an activation function $f(\cdot)$, determining the strength

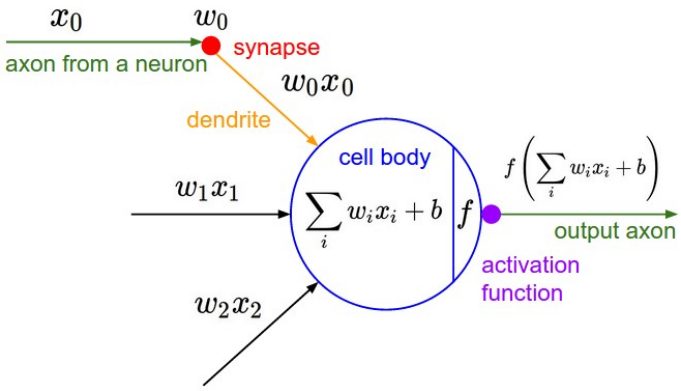


Figure 6.2: Mathematical model of a neuron. Image from Karpathy et al. (2016).

of the signal that is passed on to the next neuron (Karpathy et al., 2016). This process is schematically shown in Figure 6.2 and is analogous to what happens in a biological neuron in the central nervous system. In Figure 6.2, the terminology of the analogous biological process is also mentioned. The activation function that is applied before the output is passed on to the next layer adds a non-linearity to the model. Without it, the layers could only learn linear transformations, since the composition of linear transformations is also linear (Chollet, 2018). An activation function takes a single number and performs a non-linear mathematical operation on it, depending on the type of activation function (Karpathy et al., 2016). A commonly used activation function is the rectified linear unit (ReLU), which computes the following function:

$$f(t) = \max(0, t),$$

thresholding the activation at zero (Karpathy et al., 2016).

6.3 Convolutional neural networks

Figure 6.3 shows the difference between a regular neural network and a convolutional neural network. A neuron of a layer of a regular neural network is connected to every neuron of the previous layer. When this type of architecture is be used for image classification, an enormous number of weights would need to be learnt, since the dimensions of the input, an image, are large. For example, when the input is an image that is 200 pixels wide, 200 pixels high and has three colour channels, the input layer will need to consists of $200 \cdot 200 \cdot 3 = 120{,}000$ neurons. Since each neuron of the next layer is connected to every neuron of this layer, this would lead to neurons having 120,000 weights. With layers consisting of multiple neurons and a network consisting of multiple layers, the number of weights will quickly become enormous. A convolutional network takes advantage of the fact that the input is an image, a three-dimensional array of integers. The neurons of layers of a convolutional neural network are arranged in three dimensions. Furthermore, a neuron is

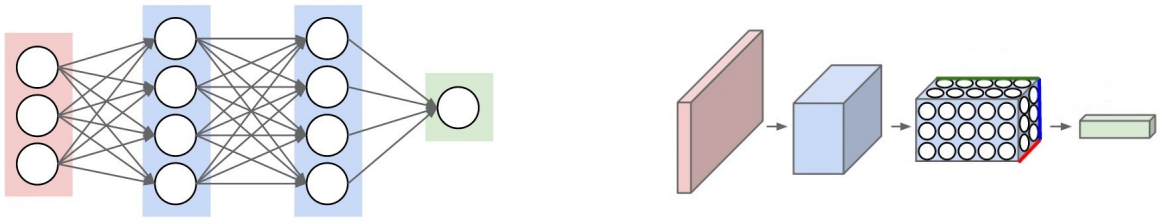


Figure 6.3: Difference in architecture between a regular neural network (left) and a convolutional network (right). Images from Karpathy et al. (2016).

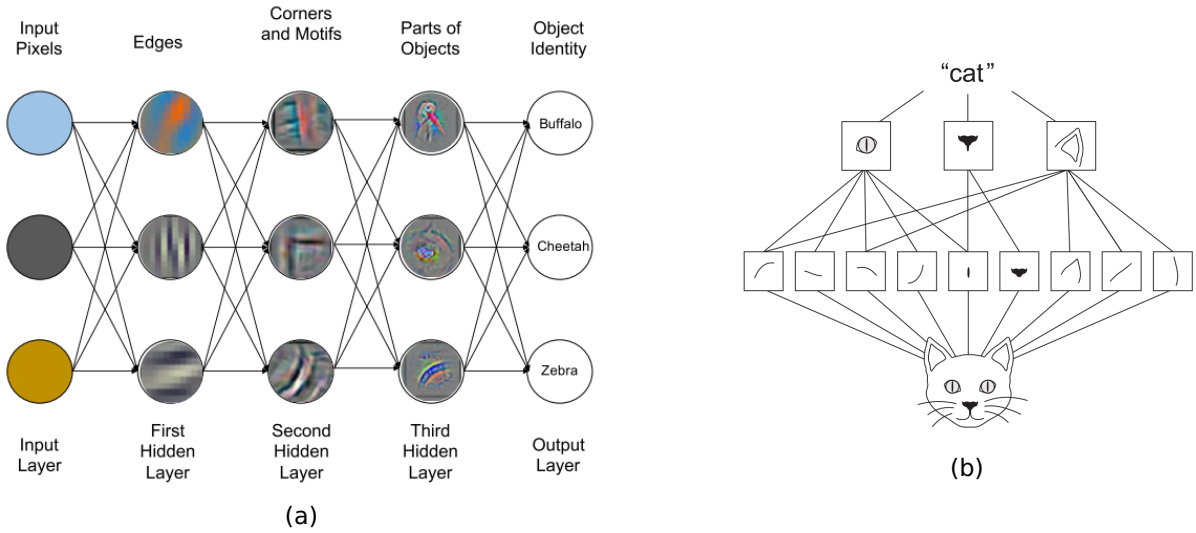


Figure 6.4: Spatial hierarchy of patterns learnt by convolution layers. Images from Norouzzadeh et al. (2017) (a) and Chollet (2018) (b).

no longer connected to all neurons of the previous layer, but only to a small region of the previous layer. When an image travels through the layers of the network, it is reduced to a single vector containing the class scores (Karpathy et al., 2016). Based on these scores, the probabilities of the possible labels, the most likely label can be determined.

Dense layers or fully-connected layers, where a neuron is connected to all neurons of the previous layer, learn global patterns in their input feature space. Convolutional layers, where a neuron is only connected to a small region of the previous layer, learn local patterns, such as edges and textures. This gives convolutional neural networks two interesting properties: they can learn spatial hierarchies in patterns and the patterns they learn are translation invariant (Chollet, 2018). Translation invariant means that a certain pattern can be recognised anywhere in the image, independent of where in the image it was learnt. This is a useful characteristic for image classification, since the object of interest may change position in the images. As explained in Section 4.2, camera trap images are often not ideal images. Animals are shown in different positions, only part of the animal is in the image, et cetera. Moreover, convolutional neural networks are able to learn spatial hierarchies of patterns. This concept is shown in Figure 6.4. A first convolution layer will learn small patterns, such as edges. A subsequent layer will be able to learn larger patterns, made of the

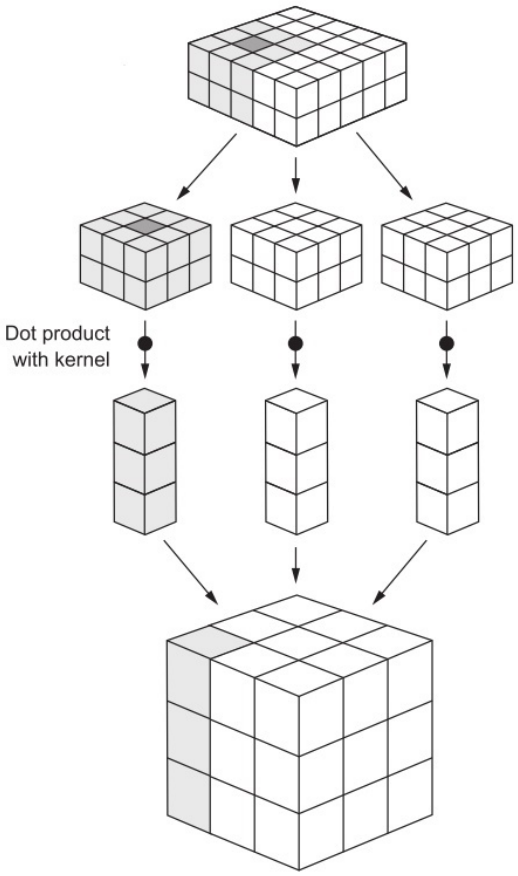


Figure 6.5: Schematic representation of how a convolution works. Image from Chollet (2018).

features of the previous layer, and so on. In this way, the network can learn increasingly complex features to classify the images (Chollet, 2018). In the example in Figure 6.4b, the low-level features are combined into more high-level features, such as the animal's eyes and ears. Based on these high-level features, the image classification model can determine what is in the picture. The process of extracting increasingly complex features is illustrated in Section 7.4.1 and Section 7.4.2.

Figure 6.5 shows schematically how a convolution works. A window with predefined dimensions, usually three by three or five by five, slides over the three-dimensional input feature map and extracts at every location a three-dimensional patch of surrounding features. Each patch is then transformed by computing the dot product between the patch and the matrix containing the weights. This matrix is called the convolution kernel in Figure 6.5. The result of this product is a one-dimensional vector. All these vectors are finally combined into a three-dimensional output feature map, preserving their spatial location (Chollet, 2018).

6.3.1 ResNet50

As mentioned earlier, the network that will be used to classify the camera trap image will not be build and trained from scratch, but a pretrained convolutional neural network will be adjusted to hierarchically classify the camera trap images. The pretrained network that will be used is ResNet, more specifically ResNet50, consisting of 50 layers. ResNet is the winning architecture of the 2016 ImageNet competition (Norouzzadeh et al., 2017) and was trained on a subset of the ImageNet dataset, containing 1.8 million images belonging to 1000 classes (Wu et al., 2016). Four other pretrained convolutional neural networks available in Keras, InceptionV3, Xception, VGG16 and VGG19, were briefly tried, but ResNet50 performed remarkably better. Also other authors working on identifying animals in camera trap images, such as Gomez et al. (2016), Norouzzadeh et al. (2017) and Gomez et al. (2017), report ResNet performing best on their data.

ResNet is short for Residual Network. Residual learning was introduced to ease the training of very deep neural networks. The depth of a network is of crucial importance for its performance, but with networks becoming deeper, a degradation problem occurred. With the network depth increasing, accuracy got saturated and adding more layers to a suitable deep model led to a higher training error. However, this degradation of the training accuracy is not caused by overfitting, but by deeper networks being more difficult to optimize (Wu et al., 2016). This problem does not appear to be caused by vanishing or exploding gradients (see Section 6.4), but by the network having a hard time learning the identity mapping for certain layers. Residual learning offers a solution to this degradation problem. Instead of letting the layers directly learn features, they learn residual functions. These residual functions are the subtraction of the feature learnt from the input of that layer. The reasoning behind this is that it may be easier for the network to learn zero mappings, where $\mathcal{M}(x) = 0$, instead of identity mapping, where $\mathcal{M}(x) = x$. This is realized by adding shortcut connections to the network that perform identity mapping. The connections directly connect the input of the n-th layer to the $(n + k)$-th layer, skipping one or more layers, as shown in Figure 6.6. When the desired underlying mapping is $\mathcal{M}(x)$, the building block (see

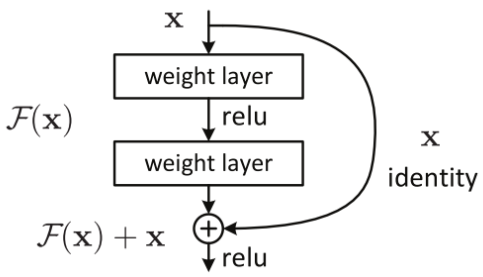


Figure 6.6: Building block for residual learning. Image from Wu et al. (2016).

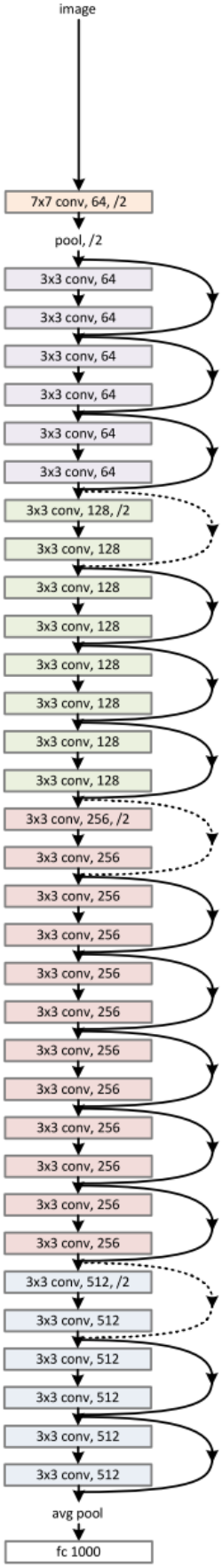


Figure 6.7: Architecture of ResNet34, a residual network with 34 parameter layers. Image from Wu et al. (2016).

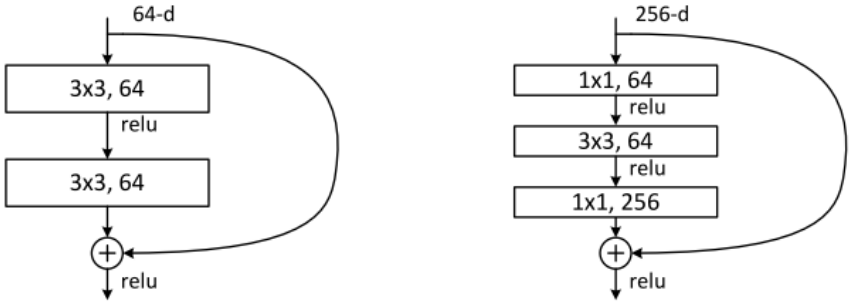


Figure 6.8: Building block for ResNet34 (left) and ResNet50 (right). Image from Wu et al. (2016).

Figure 6.6) maps $\mathcal{F}(x)$, where

$$\mathcal{F}(x) = \mathcal{M}(x) - x\,,$$

with x being the input of the first layer of the building block. The original mapping thus becomes $\mathcal{F}(x) + x$ and is reconstructed by adding the output of the shortcut connection to the output of the building block (Wu et al., 2016). Wu et al. (2016) show that is it easier to optimize the residual mapping than to optimize the original one, solving the degradation problem for very deep neural networks.

Figure 6.7 shows the architecture of ResNet34, a residual network with 34 parametrized layers. ResNet50 is constructed by replacing the building blocks of ResNet34, consisting of two convolution layers, by a new building block consisting of three convolution layers, as shown in Figure 6.8. This results in a network with 50 layers (Wu et al., 2016). A more detailed overview of the architecture of ResNet50, including the dimensions of the different layers, can be found in Dasgupta (2017).

6.4 Training a neural network

The process of training a neural network is represented in Figure 6.9. The input X, in our case a region extracted from a camera trap image, is transformed through the layers of the network, resulting in a predicted label Y'. To measure how far the output of the network Y' is from the true label Y, a loss function is used. This function computes a distance score, indicating how well the network is performing. The loss will be high if the network is doing a poor job of classifying the images and it will be low if it is doing well (Karpathy et al., 2016). This loss score can then be used as a feedback signal to adjust the weights so that the loss score is lowered. This adjustment is done by the optimizer (Karpathy et al., 2016). To find the most suitable weights, the gradient of the loss is computed, with regard to the network's weights. By updating the weights in the opposite direction from the

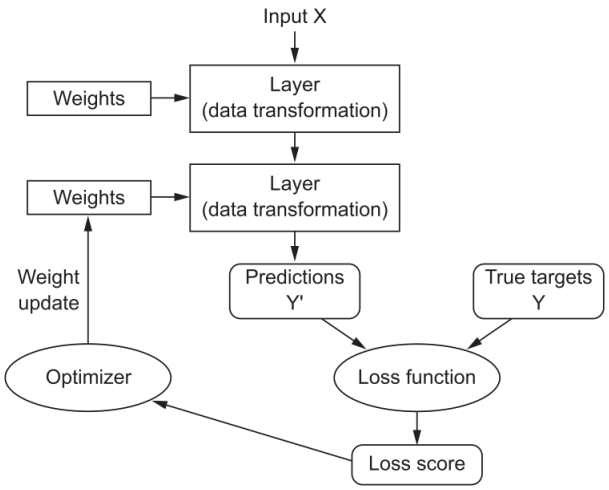


Figure 6.9: Dataflow when training a neural network. Image from Chollet (2018).

gradient, the loss decreases. To compute the gradient, advantage is taken of the fact that all operations used in the network are differentiable (Chollet, 2018). As mentioned earlier, a neural network function consists of many operations chained together. Each of these operations has a simple known derivative. By recursively applying the chain rule to this sequence of operations, the backpropagation algorithm is obtained (Karpathy et al., 2016). The chain rule is a formula used to compute the derivative of the composition of functions:

$$\frac{d}{dx}\left[f\left(g(x)\right)\right] = f'\left(g(x)\right)g'(x).$$

Backpropagation starts with the final loss value and works backwards to the bottom layers, applying the chain rule to compute the contribution of each parameter to the loss value (Chollet, 2018).

For a multiclass, single label classification problem, categorical cross entropy is recommend as a loss function (Chollet, 2018). Cross entropy is a measure from the field of information theory used to quantify the distance between probability distributions or when used as a loss function, between the ground truth distribution and the class probabilities estimated by the network (Chollet, 2018). The cross entropy between the true distribution p and the estimated distribution q is defined as:

$$\mathrm{H}(p,q) = -\sum_i p_i \log q_i$$

$$\text{where} \begin{cases} p = [p_1, \ldots, p_m] & \text{with } \sum_i p_i = 1 \text{ and } p_i \geqslant 0 \\ q = [q_1, \ldots, q_m] & \text{with } \sum_i q_i = 1 \text{ and } q_i \geqslant 0 \end{cases}$$

The true distribution is the distribution where all probability mass is on the correct class (Karpathy et al., 2016). For example **p** = [0,1,0,...,0], when the second class is the true

label of the image. As mentioned in Section 4.3, the data used to train the neural network is highly imbalanced. Therefore, the loss will be weighted, putting a higher cost on misclassifying images from rare classes and less cost on misclassifying images from frequent classes. In this way, the model will pay more attention to images from the underrepresented classes (Norouzzadeh et al., 2017). The formula used to calculate the class weights can also be found in Section 4.3.

6.5 Data augmentation

Data augmentation has been shown to be an effective, though simple technique to increase the accuracy of classification tasks (Wang and Perez, 2017). Via data augmentation, more training data is generated from existing training examples. This is done by augmenting the examples via a number of random geometric transformations, such as cropping, rotating, zooming in or out and flipping input images (Chollet, 2018). All these new variations of the original input image can also be used to train the network. This is especially useful when training a network on a small dataset. The problem that occurs when training on a small dataset, is that the model has difficulty learning to generalize to new data, which leads to overfitting. One way to reduce overfitting is using data augmentation to increase the number of training examples (Wang and Perez, 2017). At training time, the input images are augmented in such a way that the network never sees the exact same image twice (Chollet, 2018). Figure 6.10 shows some augmented images, resulting from the region that was selected after preprocessing (Figure 5.9b) of the sixth image of the example sequence used in Chapter 5. This augmentation was done by rotating, flipping and shifting the image. However, these augmented images are still heavily intercorrelated and based on the same amount of information. Therefore, data augmentation does not completely counteracts overfitting. To further reduce overfitting, other techniques can be adopted, such as adding a dropout layer to the network. Here, one randomly drops out a number of output features during training by setting them to zero (Chollet, 2018). When training the network, the gradient indicates how each weight should be adapted, given what all other weights are doing. This may lead to complex co-adaptations. By adding dropout and each time ignoring a set of output features, the output features are decorrelated (Srivastava et al., 2014).

6.6 Transfer learning

Instead of building a network from scratch and trying to optimize randomly initialised weights, a pretrained network can be used. This is a common and highly effective approach when working with a small image dataset. A pretrained network is a network that

Figure 6.10: Augmentation of the selected region after preprocessing (Figure 5.9b) of the sixth image of the example sequence used in Chapter 5 (Figure 5.2).

was previously trained on a large dataset (Chollet, 2018). A typical dataset used for this, is the ImageNet dataset, consisting of more than 14 million labelled images and over 20,000 ambiguous categories (Stanford Vision Lab, 2016). The dataset contains mostly images of animals and everyday objects. If the original dataset is large and general enough, the learnt spatial hierarchical features can be used for many different image classification problems, even when the new problem has completely different classes (Chollet, 2018). The ImageNet dataset contains many different animals classes, so the features learnt from this dataset will definitely be useful to classify wildlife in camera trap images. The pretrained network cannot directly be applied to a new classification problem, but will need some small adaptations to predict the new labels well. Two approaches will be discussed: using bottleneck features and fine-tuning.

6.6.1 Bottleneck features

A first approach is using the pretrained network to extract bottleneck features, representations learnt by the network, from the new input data. These features can then be passed on the a linear classifier, which is trained from scratch to predict the new labels, based on the features extracted by the pretrained network (Chollet, 2018). This is done by removing the top of the pretrained network, which maps the extracted features to the class scores, and replacing it by a few new layers that form a linear classifier and predict the scores for the new labels (Karpathy et al., 2016).

6.6.2 Fine-tuning

A second possible approach consists of not only replacing and training the top of the network, but also fine-tuning the weights of other layers of the pretrained network. Fine-tuning the pretrained network means slightly adjusting the weights so that the features of the pretrained model are slightly adapted to make them more relevant for the new classification problem. In principle, all layers can be fine-tuned, but to avoid overfitting the network to the new, smaller dataset, it is recommended not to change the weights of the bottom layers (Karpathy et al., 2016). Remember that the earlier layers extract local, low-level features such as edges, colours and textures, whereas layers higher up extract more high-level features, such as eyes and ears, as in the example of a cat (Figure 6.4b) (Chollet, 2018). The low-level features are more generic, whereas the more specialised features will benefit from being fine-tuned to the new problem.

6.7 Hierarchical classification

Instead of directly modelling the different output classes, the network will learn to hierarchically classify the images. Figure 6.11 shows the classification tree that is used. The explanation of the symbols used in this figure can be found in Table 6.1. First a distinction is made between blank images and images that do contain something of interest. Subsequently, the latter group is divided into images containing animals and images that do not contain animals. The images without animals are finally split in images with people or human activity and images made during the pick-up or set-up of the camera. For the images with animals, a distinction is made between birds and mammals, after which the mammals are further subdivided. Mammal species belonging to the order Rodentia (long-tailed field mouse and Eurasian red squirrel), Lagomorpha (European hare) or Eulipotyphla (Western European hedgehog) are classified as small mammals. Mammal species belonging to the order Perissodactyla (ass and horse), Carnivora (red fox, beech marten, domestic cat and domestic dog) or Artiodactyla (mouflon, western roe deer and wild boar) are classified as large mammals. Birds are not further subdivided, as discussed in Section 3.4.1.

The network first models the conditional probabilities, as shown in Figure 6.11. These probabilities are then converted to the class scores by adding a layer to the top of the network with fixed weights. When modelling the conditional probabilities, the sum of the probabilities of all children of a node must be equal to one. For example the sum of $P(A|\bar{B})$ and $P(\bar{A}|\bar{B})$ must be equal to one, but also the sum of $P(\text{Mouse}|S)$, $P(\text{Squirrel}|S)$, $P(\text{Hare}|S)$ and $P(\text{Hedgehog}|S)$ must be equal to one (see Figure 6.11). When a node has only two children, the conditional probability of one child is modelled, whereafter the conditional probability of the second child can be computed by taking the complement of probability of the first

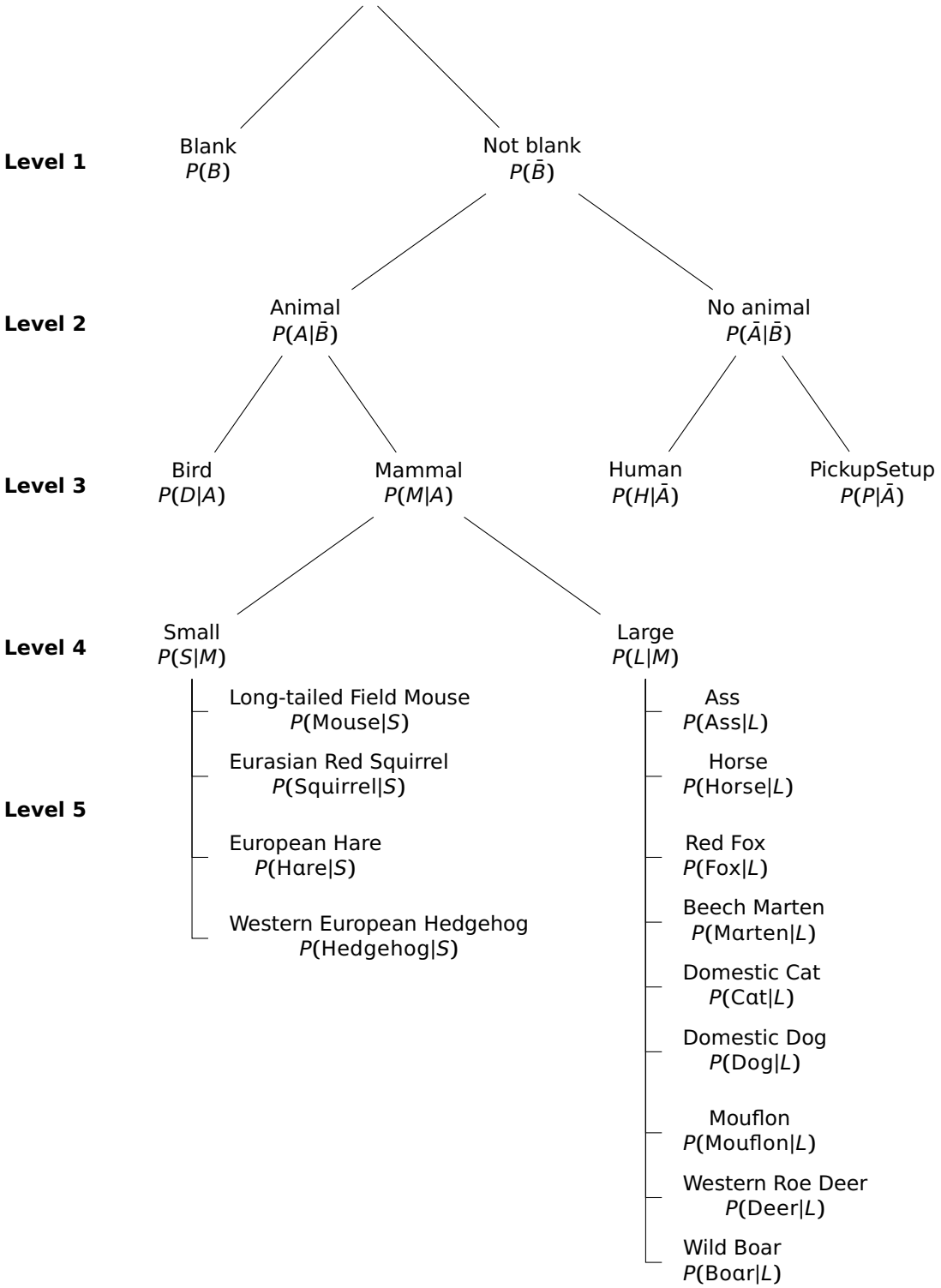


Figure 6.11: Hierarchical classification tree with the corresponding conditional probabilities. The explanation of the symbols can be found in Table 6.1.

Table 6.1: Symbols corresponding to the different hierarchical classes.

Symbol	Class
B	Blank
$\bar{B}$	Not blank
A	Animal
$\bar{A}$	No animal
D	Bird
M	Mammal
H	Human
P	PickupSetup
S	Small mammal
L	Large mammal

child, for example $P(\bar{A}|\bar{B}) = 1 - P(A|\bar{B})$. When a node has more than two children, as is the case for the small and large mammals, all children are modelled, but a softmax function is applied to make the conditional probabilities add up to one. The following formulas are used to convert the conditional probabilities to the class scores:

Level 1:

$P(B) = P(B)$

$P(\bar{B}) = 1 - P(B)$

Level 2:

$P(A) = P(A|\bar{B})\, P(\bar{B})$

$P(\bar{A}) = P(\bar{A}|\bar{B})\, P(\bar{B}) = \big(1 - P(A|\bar{B})\big)\, P(\bar{B})$

Level 3:

$P(M) = P(M|A)\, P(A)$

$P(D) = P(D|A)\, P(A) = \big(1 - P(M|A)\big)\, P(A)$

$P(H) = P(H|\bar{A})\, P(\bar{A})$

$P(P) = P(P|\bar{A})\, P(\bar{A}) = \big(1 - P(H|\bar{A})\big)\, P(\bar{A})$

Level 4:

$P(S) = P(S|M)\, P(M)$

$P(L) = P(L|M)\, P(M) = \big(1 - P(S|M)\big)\, P(M)$

Level 5:

$P(\text{Mouse}) = P(\text{Mouse}|S)\, P(S)$

$P(\text{Squirrel}) = P(\text{Squirrel}|S)\, P(S)$

$P(\text{Hare}) = P(\text{Hare}|S)\, P(S)$

$P(\text{Hedgehog}) = P(\text{Hedgehog}|S)\, P(S)$

$P(\text{Ass}) = P(\text{Ass}|L)\, P(L)$

$P(\text{Horse}) = P(\text{Horse}|L)\, P(L)$

$P(\text{Fox}) = P(\text{Fox}|L)\, P(L)$

$P(\text{Marten}) = P(\text{Marten}|L)\, P(L)$

$P(\text{Cat}) = P(\text{Cat}|L)\, P(L)$

$P(\text{Dog}) = P(\text{Dog}|L)\, P(L)$

$P(\text{Mouflon}) = P(\text{Mouflon}|L)\, P(L)$

$P(\text{Deer}) = P(\text{Deer}|L)\, P(L)$

$P(\text{Boar}) = P(\text{Boar}|L)\, P(L)$

These class scores can then be compared to the true labels in the loss function. The symbols used in the above equations, the same symbols as in Figure 6.11, are explained in Table 6.1.

The final prediction is made by passing through the classification tree, whereby at every split the branch with the highest probability is selected. In order to obtain the correct prediction at a higher level, the lower predictions must also be correct. Simply selecting the label with the highest score may lead to different results, especially since the output classes are at different levels in the classification tree. Each time we move to a lower node of the tree, an additional probability, a value smaller than or equal to one, is added to the multiplication. Therefore, the child node always has a probability that is lower than or equal to the probability of the parent node. To illustrate this, the calculation is done using the probabilities in Figure 6.12. Starting at the top of the classification tree, 'Not blank' is selected since $P(\bar{B}) > P(B)$. At the next split 'No animal' is selected since $P(\bar{A}|\bar{B}) > P(A|\bar{B})$. Finally 'Human' is selected since $P(H|\bar{A}) > (P|\bar{A})$. The final prediction is thus 'Human'. On the other hand, simply selecting the label with the highest probability would require the following calculations:

$$P(A) = P(A|\bar{B})\, P(\bar{B}) = 0.4 \cdot 0.6 = 0.24$$

$$P(\bar{A}) = P(\bar{A}|\bar{B})\, P(\bar{B}) = 0.6 \cdot 0.6 = 0.36$$

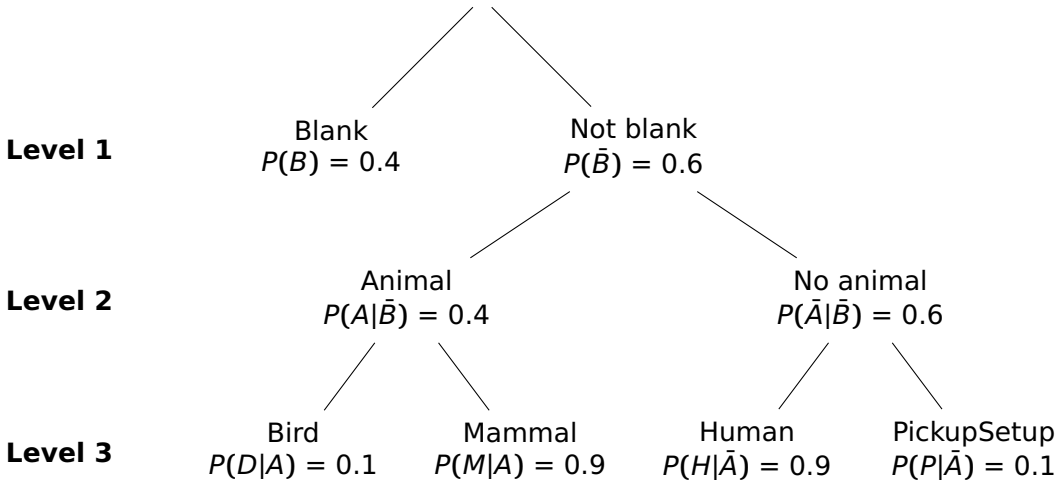


Figure 6.12: First three levels of the hierarchical classification tree with randomly chosen probabilities to illustrate how the final predictions are selected.

$$P(M) = P(M|A)\,P(A) = 0.9 \cdot 0.24 = 0.216$$
$$P(D) = P(D|A)\,P(A) = 0.1 \cdot 0.24 = 0.024$$

$$P(H) = P(H|\bar{A})\,P(\bar{A}) = 0.9 \cdot 0.36 = 0.324$$
$$P(P) = P(P|\bar{A})\,P(\bar{A}) = 0.1 \cdot 0.36 = 0.036$$

By combining the above calculations and Figure 6.12, the output probabilities are:

$$P(B) = 0.4$$
$$P(M) = 0.216$$
$$P(D) = 0.024$$
$$P(H) = 0.324$$
$$P(P) = 0.036$$

Selecting the label with the highest probability would thus result in ‘Blank’ and not ‘Human’, the correct hierarchical prediction.

By hierarchically classifying the images and modelling the conditional probabilities, a prediction can be made at every level, using the formulas above. When the network is unreliable at a certain level, rather then possibly misclassifying the image, the prediction can be limited to the level above. For example, if the trained network has difficulty discriminating between mice and hedgehogs, rather than possibly misclassifying the image, the prediction can be made one level higher in the classification tree, indicating that there is a small mammal in the image (see Figure 6.11). If small mammals are of interest for a re-

search project, one can manually classify the image in more detail, if desired. On the other hand, if the image contains nothing of interest, according to the prediction made by the neural network, the image can be discarded. In this way, the manual workload is reduced when processing camera trap images, while misclassification is mostly avoided. As shown in Table 3.1, for some animal species, very few sequences are available, making it difficult to train the network and validate its performance. For these classes, the prediction can be restricted to a level higher than the individual species level, if needed.

CHAPTER 7

RESULTS AND DISCUSSION

This chapter describes the different steps taken during training of the convolutional neural network. The performance of the final network is evaluated and analysed at the different levels of the hierarchical classification tree using confusion matrices. The confusion matrices associated with this chapter can be found in Appendix A. Finally, to gain inside in the behaviour of neural networks, different elements of what a convolutional neural network learns are visualized.

7.1 Training, validation and test data

The available data is split in a training, validation and test set. The training set consists of half of the sequences. The remaining half of the sequences is again split into two equal parts to form the validation set and the test set. Splitting the data is done randomly, but in a stratified fashion, based on the label of the sequences. In this way, the ratio of the number of sequences per class is similar for the three data sets. As shown in Table 5.1, there are only two sequences available of the Western European hedgehog, while the data needs to be split into three parts. Therefore, one sequence is assigned to the training set and the second sequence is assigned to both the validation and the test set. Table 7.1 gives an overview of the number of sequences and images of each class in the training, validation and test set. The number of images refers the the number of extracted image regions after preprocessing, not the number of camera trap images.

7.2 Training the neural network

7.2.1 Data augmentation

As explained in Section 6.5, data augmentation can be attempted to increase the network's performance. However, a disadvantage of using data augmentation in combination with bottleneck features is that the training time strongly increases. When no augmentation is applied, the bottleneck features can simply be extracted ones for all available images.

Table 7.1: Overview of the number of sequences and images of each class in the training, validation and test set. The number of images refers the the number of extracted image regions after preprocessing.

	Sequences			**Images**		
Annotation	**Train**	**Val**	**Test**	**Train**	**Val**	**Test**
Ass	4	1	2	440	400	365
Beech Marten	8	4	5	43	10	47
Bird	31	16	15	1144	556	146
Blank	792	396	396	12300	7057	6987
Domestic Cat	7	3	4	59	14	55
Domestic Dog	2	1	1	24	3	21
Eurasian Red Squirrel	3	2	1	7	8	10
European Hare	4	3	2	15	24	12
Horse	14	7	7	2348	488	1500
Human	13	6	7	235	461	535
Long-tailed Field Mouse	3	1	1	15	1	6
PickupSetup	110	55	55	39922	19627	19749
Red Fox	113	57	57	1072	688	550
Sheep	5	2	2	344	357	715
Western European Hedgehog	1	1	1	2	3	3
Western Roe Deer	632	317	316	24729	10288	10840
Wild Boar	92	46	46	4445	1929	2239
Total	1834	918	918	87144	41914	43780

The computation of the bottleneck features of all images takes about ten hours when the available graphics processing unit, a GTX 1080 Ti, is used. Thereafter, these bottleneck features can be used as an input to train the new top of the network. Training will take a few hours, depending on the number of epochs, the number of iterations over all training data. Each image thus needs to be converted to a set of bottleneck features ones. These features only depend on the pretrained network and can therefore be used multiple times when trying to find the optimal parameters for the new top. When data augmentation is applied, the network each time sees a slightly different version of the image. When enough augmentation option are provided, for example a wide zoom range, a wide shift range and a wide rotation angle, the network never even sees the exact same image twice by each time combining a different zoom, shift and rotation. However, this means that a lot of different bottleneck features will need to be extracted and that the bottleneck features cannot be stored and reused for further training since the network each time sees a different version of the images. This strongly increases the training time. The computation time is about seven times higher, whereby hours of training turn into days. In addition, augmenting the images also takes time and the more augmentation options, the more the training time increases.

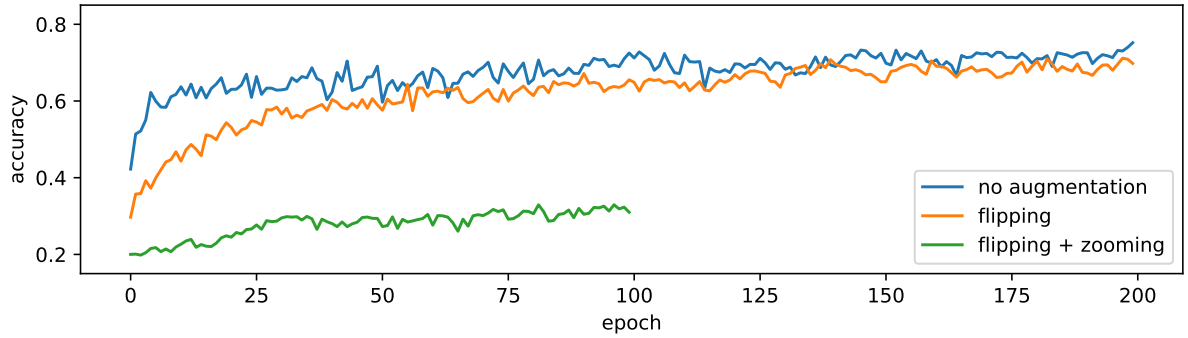


Figure 7.1: Validation accuracy without data augmentation, with horizontal flipping and with horizontal flipping combined with zooming.

Figure 7.1 shows the validation accuracy during training without data augmentation and with two data augmentation options: horizontal flipping and horizontal flipping combined with zooming. Training was ended early, after three days of training, before the accuracy reached a plateau, since it was clear that the accuracy did not benefit from adding data augmentation. Some other frequently used augmentation options such as rotating, shifting and shearing the image were briefly explored. Their results are similar to those of zooming. Figure 7.1 shows that the explored data augmentation options do not improve the network's performance. The results of horizontal flipping are similar to those without augmentation. In addition, the training time strongly increases when applying augmentation. Therefore, the further optimization of the network is done without data augmentation. Moreover, the shorter training time allows for more training options to be explored.

A possible explanation for the fact that data augmentation is not improving the performance of the network may be that the augmented images are still heavily intercorrelated, because they come from a small number of original images. Data augmentation does not produce new information. It only remixes existing information (Chollet, 2018). In addition, the resulting augmented image may not contain the object of interest, for example when zooming is applied, or the generated variations of the input data may deviate too much from the real life appearance of the objects.

7.2.2 Weighted loss function

As explained in Section 4.3, because of the heavily imbalanced data set, the network can limit its predictions to the most frequent classes, while still achieving a high level of accuracy. To conquer this, the loss is weighted, inversely proportional to the number of images per class. The weights for each class, computed using Equation 4.1 and 4.2, are reported in Table 7.2. To illustrate the importance of using weights or an other technique to conquer the imbalancedness of the data, the network was also trained without weights. Figure 7.2 shows the validation accuracy of this experiment during training. The accuracy is higher

Table 7.2: Class weights.

Class	Weight	Class	Weight
Ass	0.00261	Human	0.00489
Beech Marten	0.02673	Long-tailed Field Mouse	0.07662
Bird	0.00100	PickupSetup	0.00003
Blank	0.00009	Red Fox	0.00107
Domestic Cat	0.01948	Sheep	0.00334
Domestic Dog	0.04789	Western European Hedgehog	0.57465
Eurasian Red Squirrel	0.16418	Western Roe Deer	0.00005
European Hare	0.07662	Wild Boar	0.00026
Horse	0.00049		

when no weights are applied. However, Figure A.1 in Appendix A shows the normalized confusion matrix of the network trained without weighting. The diagonal of the normalized confusion matrix indicates the proportion of images of the different classes that is correctly classified. In Section 7.3, the performance of the network with weighting is discussed. The three classes containing the most images are 'Blank', 'PickupSetup' and 'Western Roe Deer', as shown in Table 7.1. This is reflected in the distribution of the predicted labels in the normalized confusion matrix in Figure A.1. The columns of the matrix corresponding to these classes contain a higher portion of the predictions, a result of the difference between the prior probabilities of the classes. When the network is uncertain how to classify an image of a smaller class, the best bet is to assign it to a frequently occurring class. The three aforementioned large classes have the highest accuracy, respectively 0.81, 0.84 and 0.84. The accuracy of the smaller classes is lower and for some classes even zero. The network thus has a higher overall accuracy, but does not perform well for smaller classes.

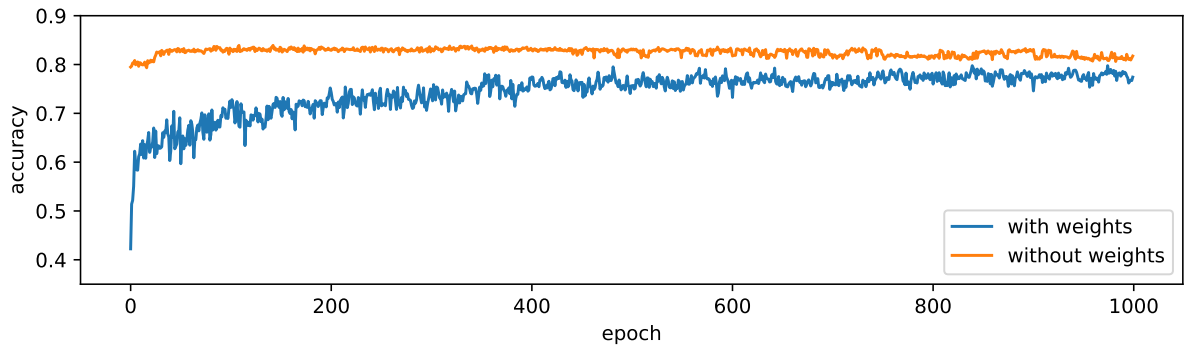


Figure 7.2: Validation accuracy during training when the loss function is weighted with the class weights reported in Table 7.2 and when the loss function is not weighted.

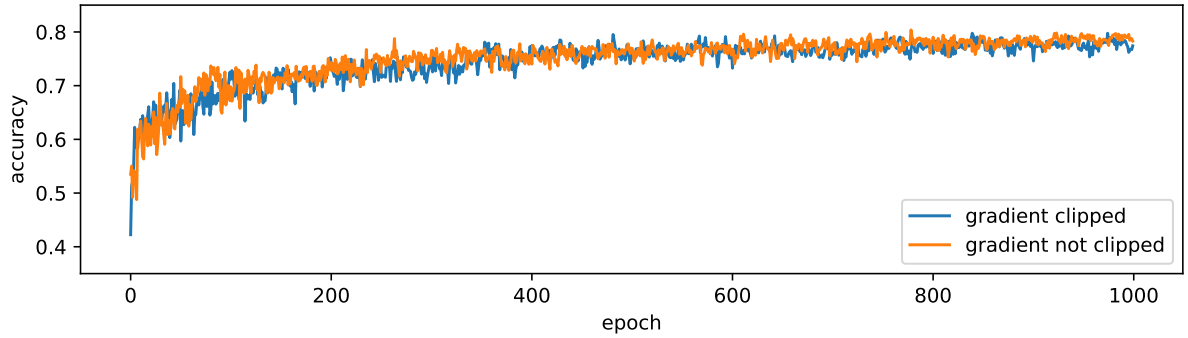


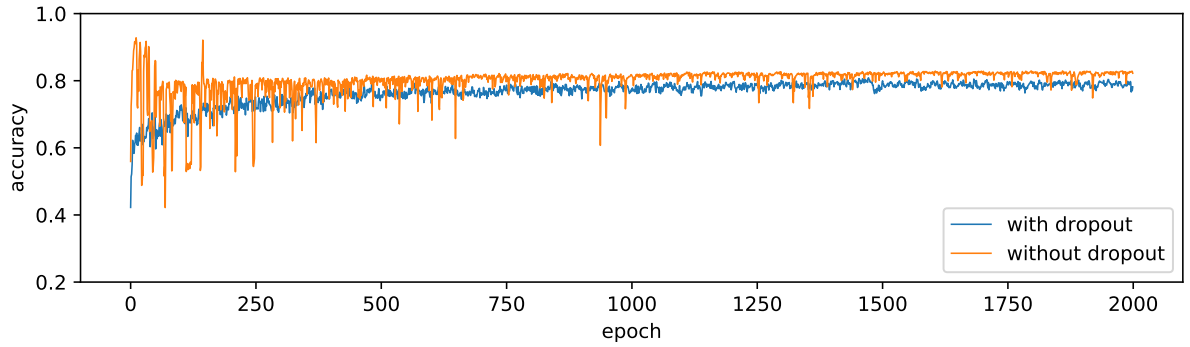
Figure 7.3: Validation accuracy during training when the gradient is clipped between -0.01 and 0.01 and when the gradient is not clipped.

7.2.3 Gradient clipping

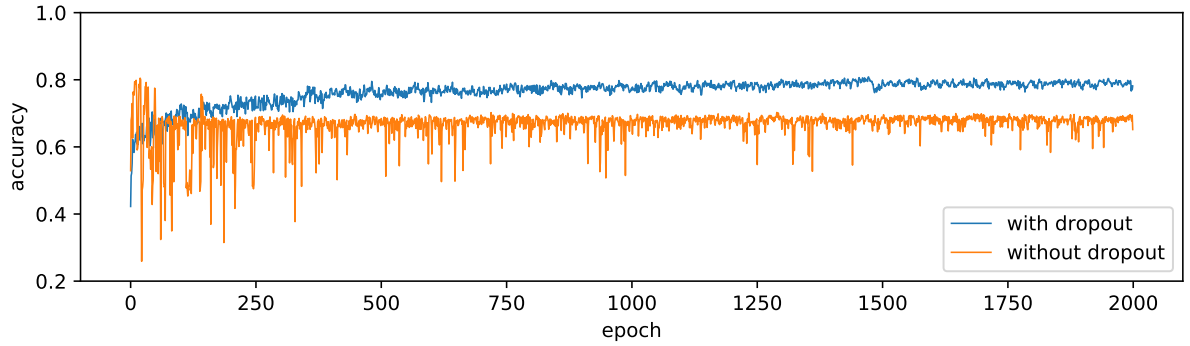
As explained in Section 4.3.2, weighting the loss could cause extremely high or low gradients when the data set is highly imbalanced, which is harmful to the learning process. Therefore, Norouzzadeh et al. (2017) suggest clipping the gradient between -0.01 and 0.01. Figure 7.3 shows the validation accuracy during training when the gradient is clipped between -0.01 and 0.01 and when the gradient is not clipped. This graph illustrates the limited imbalancedness of the data set. Weighting the loss does not cause extremely high or low gradients during training. Therefore, clipping the gradient does not affect the training process, neither positively nor negatively. The data set used by Norouzzadeh et al. (2017) however is much more imbalanced with more classes and extremer differences between the number of images per class. The extremely low and high weights needed to compensate this strong imbalancedness do give rise to extreme gradients.

7.2.4 Dropout layer

As explained in Section 6.5, a dropout layer is added to the top of the network to reduce overfitting. To illustrate the influence of a dropout layer, the network is trained without a dropout layer and with 50% dropout. Figure 7.4 shows the training and validation accuracy of this experiment. The training accuracy is slightly higher without the dropout layer, but removing the dropout layer has a negative impact on the validation accuracy. Hence, without the dropout layer, the network is indeed overfitting on the training data. It models the training data slightly better, but loses its ability to generalize to new data, resulting in a lower performance on the validation data.



(a) Training



(b) Validation

Figure 7.4: Training and validation accuracy of the network with and without a 50% dropout layer.

7.2.5 Fine-tuning

Once the new top is trained, it is possible to fine-tune the layers of the convolutional base, the pretrained network. It is important to first train the new top before starting to fine-tune the weights of the pretrained network. Otherwise, fine-tuning could destroy the features learnt by the pretrained network because the error signal propagating through the network during training is too large when the weights of the top are not yet optimized (Chollet, 2018). As explained in Section 6.6.2, when working with small data sets, it is recommended to only fine-tune the top layers of the pretrained network. Figure 7.5 shows the validation accuracy of the network before fine-tuning and during fine-tuning of the last convolutional building block of the pretrained base. This is just a small experiment to investigate the possible benefit of fine-tuning. Due to the very large training times for fine-tuning, the full potential of fine-tuning is not yet explored. Using the available GPU, training for just 100 epochs takes five days. Exploring the full potential of fine-tuning and exploring which layers need fine-tuning to get the best result without overfitting would take months and is therefore left for future work.

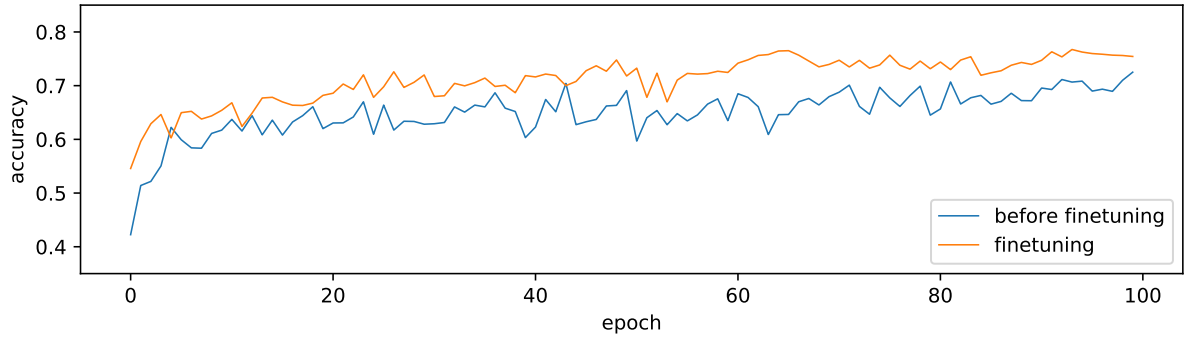


Figure 7.5: Validation accuracy before fine-tuning and during fine-tuning.

7.2.6 Final neural network

The final network contains a dropout layer and is trained without data augmentation during 3000 epochs. Only the new top is trained. The convolutional base is not additionally fine-tuned. The loss is weighted with class weights to correct for the imbalancedness of the data. Clipping the gradient is not necessary since the data set is not that strongly imbalanced that it causes extreme gradients.

Figure 7.6 shows the training and validation accuracy of the final model. The overall accuracy on the test images is 0.80. The performance of the network is discussed in detail in Section 7.3. Figure 7.7 summarises the architecture of the final model, consisting of the base of ResNet50 with a new top. The two top layers, 'softmax_layer' and 'cond_layer' are added to allow hierarchical classification. As explained in Section 6.7, a softmax function is applied to make the conditional probabilities of the child nodes add up to one. The final layer, 'cond_layer', converts the conditional probabilities to the final output probabilities of the classes.

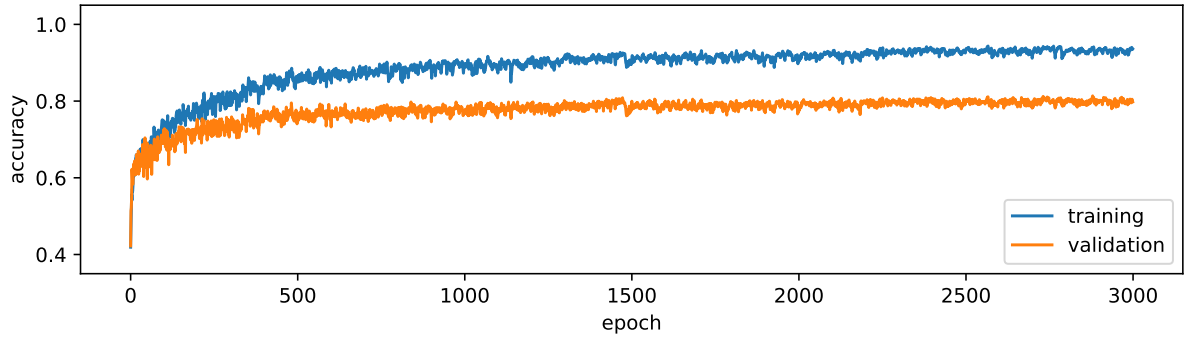


Figure 7.6: Training and validation accuracy of the final model.

Layer (type)	Output Shape	Parameters
ResNet50 (Model)	(None, 1, 2, 2048)	23587712
Flatten_1 (Flatten)	(None, 4096)	0
Dense_1 (Dense)	(None, 256)	1048832
Dropout_1 (Dropout)	(None, 256)	0
Dense_2 (Dense)	(None, 18)	4626
Softmax_layer (Lambda)	(None, 18)	0
Cond_layer (Lambda)	(None, 17)	0

Total parameters: 24,641,170
Trainable parameters: 1,053,458
Non-trainable parameters: 23,587,712

Figure 7.7: Summary of the architecture of the final model.

7.3 Neural network performance

7.3.1 Predicting sequences

As explained in Section 4.4, the predicted labels of the individual images of a sequence need to be aggregated to one label for the whole sequence, since the camera trap images are annotated at sequence level. Two methods are explored: selecting the most frequently predicted label and averaging the top-k predictions of all images of the sequence. In both methods, the classification is done hierarchically, as explained in Section 6.7. Starting at the top of the classification tree, the labels of the individual images are combine to determine the predicted label of the sequence at the first level of the tree. If the prediction at the first level is 'Not blank', the same process is repeated at the second level of the tree and so on. The way the label of the sequence is determined, depends on the method that is being used. In method one, the most frequently predicted label is selected. In method two, the probabilities of the top-k predictions of every image are averaged over all images of the sequences. The label of the sequences is then the prediction with the highest average probability. The number of predictions k that is taken into account, depends on the available nodes at a certain level. At level 1 to 4, each node has two child nodes so the top-2 predictions are averaged. At level 5, the node 'Small mammal' has four children so the top-4 is used. For the child nodes of 'Large mammal', the top-5 is used. Table 7.3 summarizes the results of both methods, as well as the results for the individual images.

Table 7.3: Accuracy on the validation images and sequences (most frequent prediction and top-*k* prediction) with the hierarchical predictions limited to the different levels.

	Images	**Sequence labelling method**	
		Frequency	**Top-k**
Level 1	0.925	0.922	0.923
Level 2	0.872	0.912	0.913
Level 3	0.862	0.902	0.903
Level 4	0.862	0.894	0.895
Level 5	0.796	0.730	0.816

Table 7.4: Accuracy on the test sequences by averaging the top-*k* predictions with the hierarchical predictions limited to the different levels.

Level 1	0.920
Level 2	0.899
Level 3	0.886
Level 4	0.882
Level 5	0.791

From this table we can conclude that averaging the top-*k* predictions leads to a higher accuracy at all levels. This method will be used to compute the label of the sequences. Table 7.3 also shows that the accuracy of the sequences, mostly based on multiple images, is slightly higher than the accuracy of the individual images.

The selected method, averaging the top-*k* prediction, is applied to the test images to get the accuracy of the network on the test sequences. The results are shown in Table 7.4.

7.3.2 Confusion matrices

The confusion matrices at the different levels of the classification tree provide insight into which classes the network correctly predicts and which not. The confusion matrices and normalized confusion matrices can be found in Appendix A. The label ‘Other’ comprises the output classes located at a higher level. At level 2 and 3, ‘Other’ includes ‘Blank’. At level 4 and 5, ‘Other’ includes ‘Bird’,‘Human’ and ‘PickupSetup’.

Figure A.2 shows the confusion matrices at the first level of the classification tree. 83% of the empty sequenced is classified correctly (true positive). In addition, less than 1% of the sequences that are in reality not empty are labelled ‘Blank’ (false positive). This combination of many true positive and few false positive predictions makes the network’s predictions very suitable to remove the empty images from the data set. Because of the low false positive rate, only a very small portion the data will get lost. On the other, because of the high true positive rate, a lot of the empty images are removed. The predictions at

the first level of the classification tree are thus very suitable to do a first clean-up of the data. 49% of the sequences in the data set are empty, so automatically removing these sequences is a first step to reduce the manual workload.

Figure A.3 shows the confusion matrices at the second level of the classification tree. Here, a distinction is made between non-empty sequences containing animals and sequences without animals. 96% of the sequences containing animals and 92% of the non-empty sequences containing no animal are classified correctly. The predictions at this level can also be used to clean up the data since only sequences containing animals are of interest.

Figure A.4 shows the confusion matrices at the third level of the classification tree. Sequences containing animals are subdivided into birds and mammals, while the non-empty sequences without an animal are subdivided into sequences containing humans and sequences made during the pick-up and set-up of the camera. Figure A.4 shows that the network has a hard time discriminating between 'Human'-sequences and 'PickupSetup'-sequences. This was to be expected as the only difference between the humans in these sequences is that the humans in 'Human'-sequences are passers-by, while the humans in 'PickupSetup'-sequences are involved in the camera trapping project. Therefore, it is recommended to restrict the hierarchical prediction at 'No animal' and not further subdivide the sequences. This should not be a problem since all 'No animal'-sequences need to be removed from the data set. Figure A.4 also shows that the accuracy for sequences containing birds is low. The data set contains only few sequences with birds and as explained in Section 3.4.1 these sequences are often not ideal and often only the first image(s) of the sequence contain(s) the bird. The accuracy of the mammals however is 96%. Generally, one is not interested in bird species if the configuration of the camera trapping framework is optimized to capture mammals. Classifying the birds to species level will therefore not be the main objective. However, improving the ability of the network to recognise birds will prevent them for being misclassified as mammal species and contaminate those sequences.

Figure A.5 shows the confusion matrices at the fourth level of the classification tree, where the mammal species are subdivided into small and large mammals. 97% of the sequences containing large mammals are classified correctly. The accuracy for the small mammals however is zero. None of the sequences containing small mammals are classified correctly. The data set contains only few sequences with small mammals making it difficult for the neural network to learn their appearance. As a result of the small number of sequences, the test set consists of only five sequences with small mammals, making the performance measures also unreliable.

Finally, Figure A.6 and Figure A.7 show the confusion matrix and the normalized confusion matrix at the fifth level of the classification tree. As can be seen in Figure A.6, most of these

output classes contain only a few sequences, making it difficult to draw reliable conclusions about their accuracy. Figure A.6 shows that for some of the classes a high accuracy is achieved, but for the class containing only a few sequences, this may have been a lucky shot. The only animal species for which a sufficient number of sequences is available and a good performance is achieved is Western roe deer. 88% of the sequences containing deer are classified correctly. However, a lot of other animal species are misclassified as 'Deer'. Therefore, the network can be used to extract the sequences containing deer, but the selected sequences would need manual clean-up to remove the sequences that are wrongly classified as 'Deer'. The predictions at the final level of the classification tree need further improvement, which requires more data. To have an overview of the accuracy of all classes, independent of their level in the classification tree, Figure A.8 and Figure A.9 are added to Appendix A, showing the confusion matrix and the normalized confusion matrix of all output classes.

7.3.3 Examples of misclassified and correctly classified images

Figure 7.8 shows four images that are labelled incorrectly by the network. Figure 7.8a contains a blurred deer occluded by a bush and is wrongly labelled as 'Fox'. Figure 7.8b and Figure 7.8c show a part of a boar and are respectively labelled 'Fox' and 'Blank'. A convolutional neural network is specialized in certain parts, features of animals to recognize the species. When that part of the animal is not in the images, the network is not able to recognize the species. To be able to correctly classify the image in Figure 7.8b, the network should have a very good notion of the difference between the fur pattern of a fox and a boar. Figure 7.8d contains a fox occluded by a bush and is wrongly labelled as 'Blank'. Because of the occlusion, the network is not able to recognize the fox's head and identify the animal. Figure 7.9 on the other hand shows two images that are labelled correctly, while one would expect them to be difficult to classify. Figure 7.9a shows a blurry image of a deer and in Figure 7.9b the deer is positioned is such a way that only its back is visible. Nevertheless, the network is able to correctly classify the images.

7.4 Visualization of what a convolutional neural network learns

By visualizing different elements of what a convolutional neural network learns, insight into the internal operations and the behaviour of these complex models can be gained (Zeiler and Fergus, 2014). A wide range of techniques have been developed to visualize representations learned by convolutional neural networks. Three of these techniques are applied in

(a) **Deer** - Fox

(b) **Boar** - Fox

(c) **Boar** - Blank

(d) **Fox** - Blank

Figure 7.8: Examples of images that are labelled incorrectly by the network. The true label is indicated in bold.

(a) Deer

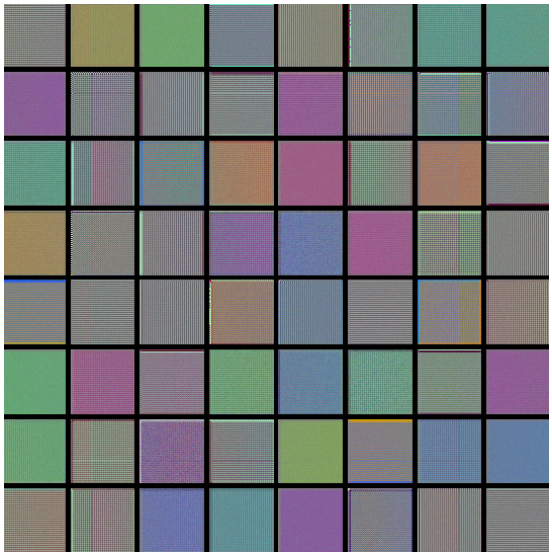
(b) Deer

Figure 7.9: Examples of images that are labelled correctly by the network.

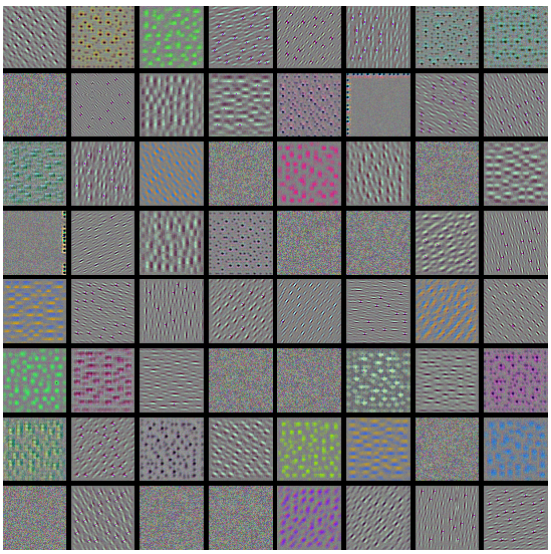
this section: visualizing filters, visualizing intermediate activations and heat maps of class activation.

7.4.1 Filters

As explained in Section 6.3 and illustrated in Figure 6.5, a convolution operation extracts patches from the input features map. The same transformation is applied to all of these patches, by means of the convolution kernel, resulting in a three-dimensional output feature map (Chollet, 2018). In the original image, the depth stands for the different channels of the image, representing the different colours. In the output feature map, the depth no longer stands for different colours, but for different filters (Chollet, 2018). As explained



(a) conv1



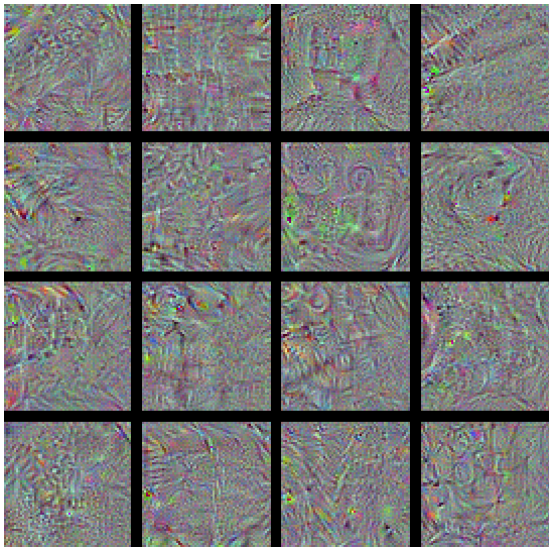
(b) pool1

Figure 7.10: Filter patterns of two early layers of ResNet50.

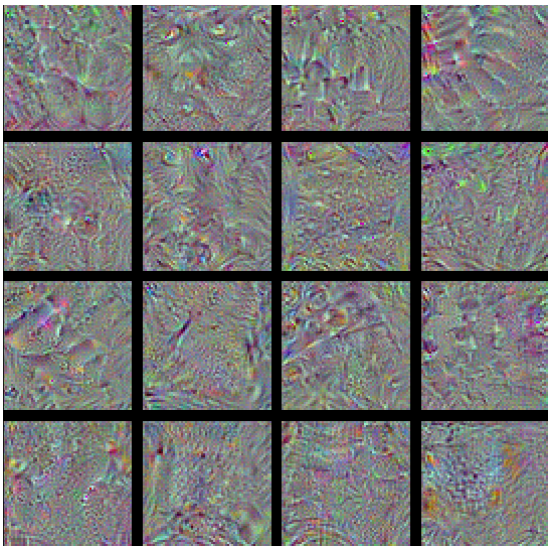
earlier, these filters encode specific aspects of the input data. Early layers learn small patterns, while subsequent layers learn larger patterns, made from the features of the previous layers. By visualizing these filters, one can find out what visual pattern each filter is receptive to.

To find the pattern that a filter is meant to respond to, a randomly generated input image is adapted in such a way that the response of the filter to that image is maximized. This is done by applying gradient ascent to the values of the input image (Chollet, 2018). This process is similar to minimizing the loss function by changing the weights in the opposite direction of the gradient, when training the network, but now the loss function will be maximized by changing the pixel values of the image in the direction of the gradient. Finally, the resulting values are postprocessed to turn them into a displayable image, since the resulting pixel values are not necessarily positive integer values.

Figure 7.10 displays the filter patterns of two early layers of the neural network ResNet50. Figure 7.10a displays the first convolutional layer at the bottom of the network and Figure 7.10b displays the subsequent max pooling layer (see Figure 6.7). These layers encode simple patterns such as edges and colours and subsequently simple textures made from combinations of these edges and colours. Figure 7.11 displays the first 16 filter patterns of two higher layers of the neural network ResNet50. These layers are part of the last convolutional building block. The patterns are much more complicated and start to look more like natural patterns, such as leaves, fur patterns and eyes.



(a) res5c_branch2b



(b) res5c

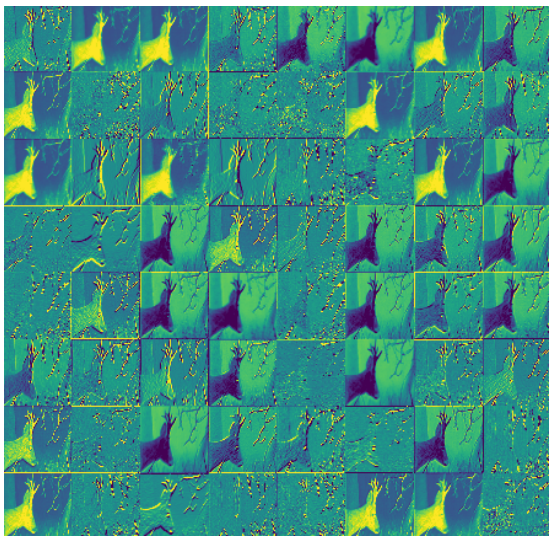
Figure 7.11: First 16 filter patterns of two higher layers of ResNet50.

7.4.2 Intermediate activations

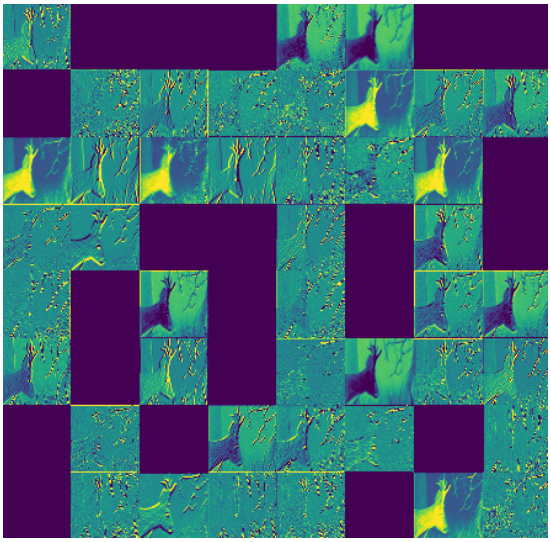
Another way to represent what neural networks learn, is visualizing the intermediate outputs. This helps to understand how successive layers transform their input. Given a certain input image, the feature maps that are produced by the layers of the network can be displayed. This shows how the input is decomposed into different filters (Chollet, 2018). In Section 7.4.1, the patterns to which the filters respond maximally are visualized. Here, the results of applying the filters to an input image, the extracted features, are visualized. The input image that is used, is shown in Figure 7.12. This image results from applying the preprocessing steps described in Chapter 5 on a camera trap image.

Figure 7.13 shows the intermediate activations of two early layers of ResNet50 for the input image. These layers are the same layers as in Figure 7.10, the first convolutional layer at the bottom of the network and the subsequent max pooling layer (see Figure 6.7). The intermediate activations clearly show that the early layers act as an edge detector. The patterns of the layer displayed in Figure 7.10b are more complex then the patterns

Figure 7.12: Input image used to visualize intermediate activations.

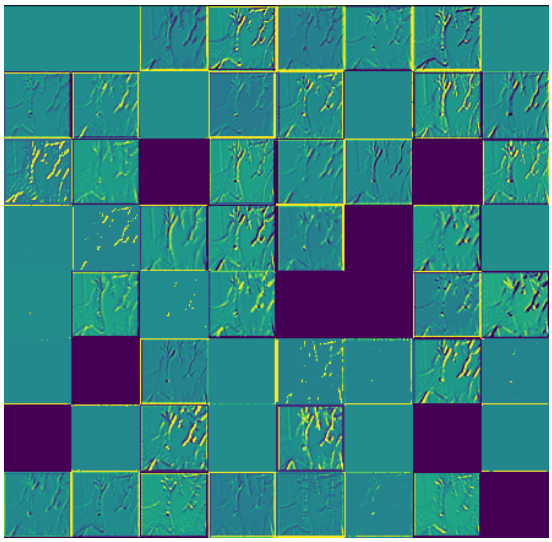


(a) conv1

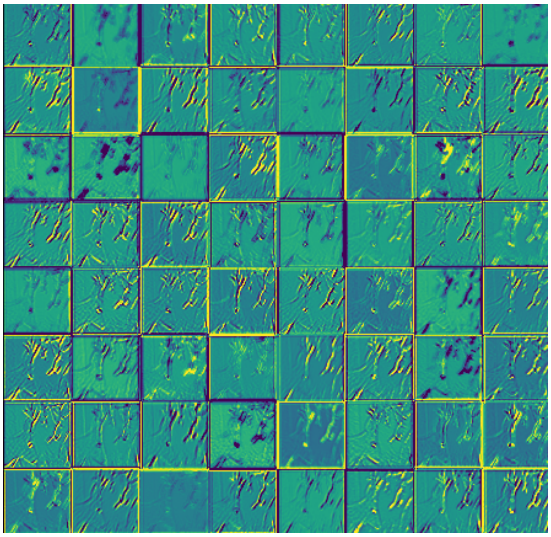


(b) pool1

Figure 7.13: Intermediate activations of two early layers of ResNet50 for the input image in Figure 7.12.



(a) res2c_branch2b



(b) res2c

Figure 7.14: Intermediate activations of two higher layers of ResNet50 for the input image in Figure 7.12.

in Figure 7.10a. This results in more blank intermediate activations because the pattern encoded by the filter is not found in the input image. However, the filters of the early layers that are activated, retain almost all information present in the input image. This can also be seen in Figure 7.13, where most of the area of the filter is activated, if the filter is activated. Figure 7.14 shows the intermediate activations of two higher layers of ResNet50, belonging to the third convolutional building block. Here, the activations begin to encode higher-level features such as eyes and the antlers of the roe deer.

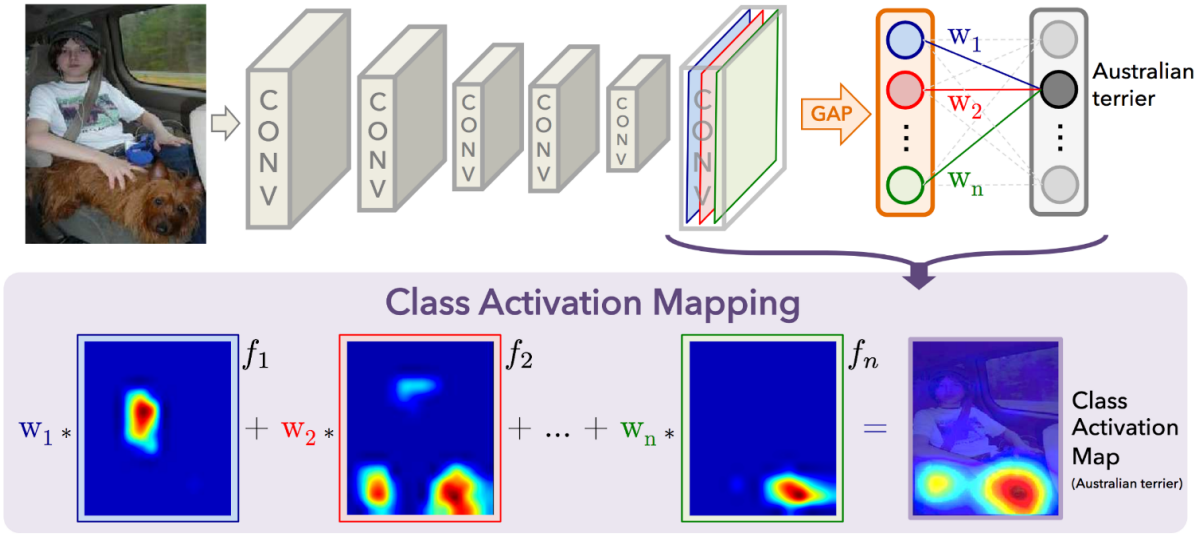
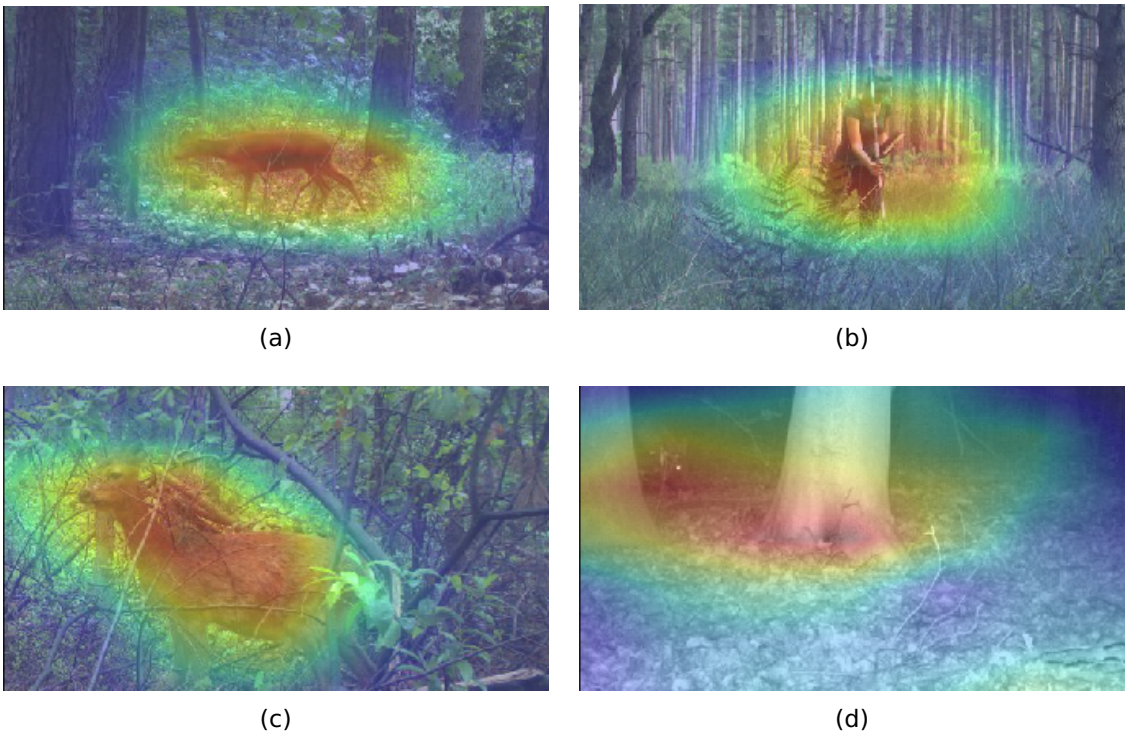


Figure 7.15: Schematic representation of class activation mapping, based on a global average pooling (GAP) layer. Image from Zhou et al. (2016).

7.4.3 Heat maps of class activation

A third way to help understand what neural networks learn is plotting heat maps of class activation. These maps indicate which parts of an input image led the network to its prediction (Chollet, 2018). In 2015, Zhou et al. (2016) showed that convolutional neural networks with global average pooling layers trained for a classification task, can also be used for object localization. A convolutional neural network with a global average pooling layer therefore cannot only predict what object is contained in the image, but can also indicate the location of that object. As shown in Figure 6.7, the top of the convolutional neural network contains a global average pooling layer. This layer reduces the dimensions of the output of the last convolutional layer, a three-dimensional matrix containing the extracted feature maps. Each feature map is reduced to a single number by taking the spatial average of all values of that feature map. A weighted sum of these average features is used to generate the final output (Cook, 2017).

Each of the feature maps in the last convolutional layer preceding the global average pooling layer acts as a detector for a different pattern in the image. To find out the location of the object, these detected patterns need to be transformed to detected objects. To do so, a class activation map is computed. This map indicates the discriminative image regions used by the network to identify a class (Cook, 2017). A weighted sum of the feature maps of the last convolutional layer of the network is computed to generate these class activation maps (Zhou et al., 2016). This process is schematically shown in Figure 7.15. Each node of the global average pooling layer corresponds to a different feature map. The contribution of these feature maps to the predicted object class is expressed by the weights of the connection between the final layer and the global average pooling layer. Every node of this final



(a) (b)

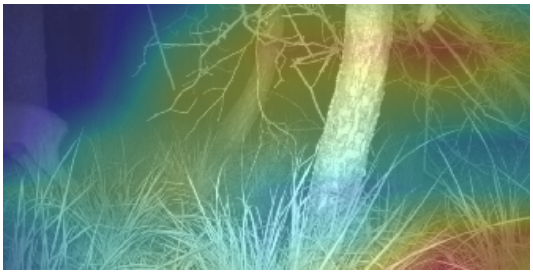
(c) (d)

Figure 7.16: Class activation maps superimposed on the original input images for object localization.

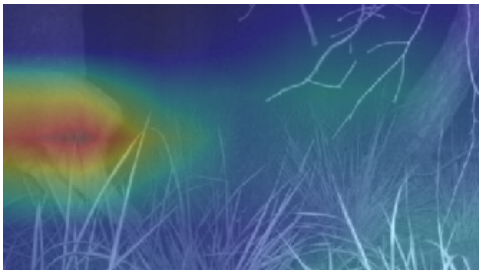
layer is connected to all the nodes of the global average pooling layer. To obtain the class activation map, the contribution of each of the feature maps, the detected patterns, to the final prediction is computed (Cook, 2017). This is done by multiplying the activation of the feature maps of the last convolutional layer with the corresponding weights of the connection between the global average pooling layer and the layer generating the final output. In Figure 7.15, the features maps f_n are multiplied with the weights w_n of the connection between the global average pooling layer and the final layer generating the predictions, after which they are summed to form the activation map (Zhou et al., 2016). As shown in Figure 7.15, the first filter does react to the boy's face, but its importance is reduced by the corresponding weight, resulting in the face not being important for the final prediction. The region of the image containing the dog however is of importance for the final prediction.

Figure 7.16 shows the result of applying this technique to four images, extracted from camera trap images by applying the preprocessing steps. In Figure 7.16a, Figure 7.16b and Figure 7.16c, the network is able to detect the animals and human. As explained in Section 6.6, ResNet50 was trained on the ImageNet dataset, a data set containing images of many different animal classes, amongst other classes. Therefore, the features learnt by this network are also useful to classify camera trap images. The network correctly predict the label of the images in Figure 7.16a, Figure 7.16b and Figure 7.16c, respectively 'Western Roe Deer', 'PickupSetup' and 'Western Roe Deer'. In Figure 7.16d, the network has difficulty

(a)



(b)



(c)

Figure 7.17: Original camera trap image on which the region of interest determined by the preprocessing step is indicated (a), class activation map of the full camera trap image (b) and class activation map of the extracted image region (c).

detecting the small wild boar in the upper left corner. It seems distracted by the large trunks in the image. However, the predicted label is still correct: 'Wild Boar'.

To highlight the added value of applying the preprocessing steps, Figure 7.17 shows the class activation map of a full camera trap image and of the image region extracted from this camera trap image by applying the preprocessing steps. In Figure 7.17b, the full camera trap image, the network is not able to detect the hindquarters of the Western roe deer, while in Figure 7.17c, the network is able to localize the deer.

CHAPTER 8

CONCLUSIONS AND FUTURE PERSPECTIVES

The convolutional neural network is able to identify 83% of the empty sequences that were not yet removed during preprocessing. At the same time, less than 1% of the sequences that are not empty are labelled 'Blank'. This result shows that the neural network is suitable to do a first clean-up of the data by removing the empty sequences. This is a big improvement to reduce the manual workload since about half of the sequences are blank and deciding that there is actually no animal in the image, is typically the most time consuming part of manually processing the images. The main objective of this research was to remove images containing passers-by so that the data can be made available for citizen science. The network successfully recognises 92% of the non-empty sequences that where triggered by human activity. The neural network is not yet able to reach a high performance when classifying the images to species level, especially for the many smaller classes. However, the hierarchic nature of the classification method allows us to limit the prediction to a higher level, achieving a better performance. Since removing the sequences containing humans can be automated, citizen science can be a helpful tool to identify the animal species. Moreover, in this way, more labelled data is generated so that the performance of the network can be improved. Collecting sufficient labelled data remains a persistent challenge in computer vision applications.

To select the regions of interest in the camera trap images, the data is preprocessed. This has been shown to be a useful step in the classification process. Moreover, preprocessing already removes some of the blank images and images made during the pick-up or set-up of the camera, containing nothing of interest. However, still some of the sequences that do contain animals get lost during preprocessing. Further improvement of the preprocessing algorithm could solve this problem. Another option is integrating the preprocessing steps into the neural network. This would require new layers to be added to the bottom of the pretrained network or building and training a neural network from scratch. More data would allow the exploration of these options.

A more thorough evaluation of data augmentation techniques and fine-tuning the network could lead to further improvement of the performance. This is not yet done due to hardware limitation, resulting in very large training times. Furthermore, the network now classifies individual images, whereafter these predictions are aggregated to label the sequences. It should be possible to train a neural network to directly classify sequences, but this causes challenges related to sequences with different numbers of images and larger neural network sizes that not have been solved. Finally, classifying sequences which contain multiple animal species is an even bigger challenge that needs to be addressed in the future.

BIBLIOGRAPHY

Bonney, R., Cooper, C. B., Dickinson, J., Kelling, S., Phillips, T., Rosenberg, K. V., and Shirk, J. (2009). Citizen science: A developing tool for expanding science knowledge and scientific literacy. *BioScience*, 59(11):977–984.

Burton, A. C., Neilson, E., Moreira, D., Ladle, A., Steenweg, R., Fisher, J. T., Bayne, E., and Boutin, S. (2015). Wildlife camera trapping: A review and recommendations for linking surveys to ecological processes. *Journal of Applied Ecology*, 52:675–685.

Chollet, F. (2018). *Deep Learning with Python*. Manning.

Cohn, J. P. (2008). Citizen science: Can volunteers do real research? *BioScience*, 58(3):192–197.

Cook, A. (2017). Global average pooling layers for object localization. Retrieved from https://alexisbcook.github.io/2017/global-average-pooling-layers-for-object-localization.

Dasgupta, S. (2017). ResNet-50. Retrieved from http://ethereon.github.io/netscope/#/gist/db945b393d40bfa26006.

Foster, R. J. and Harmsen, B. J. (2012). A critique of density estimation from camera-trap data. *Journal of Wildlife Management*, 76(2):224–236.

Gomez, A., Diez, G., Salazar, A., and Diaz, A. (2016). Animal identification in low quality camera-trap images using very deep convolutional neural networks and confidence thresholds. In *Advances in Visual Computing*, pages 747–756. Springer.

Gomez, A., Salazar, A., and Vargas, F. (2017). Towards automatic wild animal monitoring: Identification of animal species in camera-trap images using very deep convolutional neural networks. *Ecological Informatics*, 41:24–32.

Haralick, R. M. and Shapiro, L. G. (1985). Image segmentation techniques. *Computer Vision, Graphics, and Image Processing*, 29:100–132.

Haralick, R. M., Sternberg, S. R., and Zhuang, X. (1987). Image analysis using mathematical morphology. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(4):532–550.

Hasselt University (2015). Year report field research centre Hasselt University. Technical report.

Hedges, L., Lam, W. Y., Campos-Arceiz, A., Rayan, D. M., Laurance, W. F., Latham, C. J., Saaban, S., and Clements, G. R. (2015). Melanistic leopards reveal their spots: Infrared camera traps provide a population density estimate of leopards in malaysia. *Journal of Wildlife Management*, 79(5):846–853.

Henschel, P., Azani, D., Burton, C., Malanda, G. U. Y., Saidu, Y., Sam, M., and Hunter, L. (2010). Lion status updates from five range countries in West and Central Africa. *Cat News*, 52:34–39.

Huber, P. J. (2011). Robust Statistics. In *International Encyclopedia of Statistical Science*, pages 1248–1251. Springer.

Jeanloz, S., Lizin, S., Beenaerts, N., Brouwer, R., Van Passel, S., and Witters, N. (2016). Towards a more structured selection process for attributes and levels in choice experiments: A study in a Belgian protected area. *Ecosystem Services*, 18:45–57.

Karpathy, A., Li, F.-F., and Johnson, J. (2016). CS231n: Convolutional Neural Networks for Visual Recognition. Retrieved from http://cs231n.github.io.

Khatib, F., Dimaio, F., Cooper, S., Kazmierczyk, M., Gilski, M., Krzywda, S., Zabranska, H., Pichova, I., Thompson, J., Popović, Z., Jaskolski, M., and Baker, D. (2010). Crystal structure of a monomeric retroviral protease solved by protein folding game players. *Nature Structural and Molecular Biology*, 18(10):1175–1177.

Lambrechts, J., Boers, K., Keulemans, G., Jacobs, M., Moens, L., Renders, M., and Willems, W. (2013). Monitoring ecoduct 'De Warande' over de N25 in Meerdaalwoud (Bierbeek).

Li, C., Zhao, C., and Fan, P. F. (2015). White-cheeked macaque (Macaca leucogenys): A new macaque species from Medog, southeastern Tibet. *American Journal of Primatology*, 77:753–766.

López, V., Fernández, A., García, S., Palade, V., and Herrera, F. (2013). An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information Sciences*, 250:113–141.

MacKenzie, D. I., Nichols, J. D., Lachman, G. B., Droege, S., Royle, A. A., and Langtimm, C. A. (2002). Estimating site occupancy rates when detection probabilities are less than one. *Ecology*, 83(8):2248–2255.

McShea, W. J., Forrester, T., Costello, R., He, Z., and Kays, R. (2016). Volunteer-run cameras as distributed sensors for macrosystem mammal research. *Landscape Ecology*, 31(1):55–66.

Natuurpunt (2018). Het grote vogelweekend van Natuurpunt. Retrieved from https://vogelweekend.natuurpunt.be.

Norouzzadeh, M. S., Nguyen, A., Kosmala, M., Swanson, A., Packer, C., and Clune, J. (2017). Automatically identifying wild animals in camera trap images with deep learning.

Privacycommissie (2018). Camera's voor verschillende doeleinden: Andere privacyregels. Retrieved from https://www.privacycommission.be/nl/cameras.

Reconyx (2013). HyperFire instruction manual.

Reed, J., Raddick, M. J., Lardner, A., and Carney, K. (2013). An exploratory factor analysis of motivations for participating in Zooniverse, a collection of virtual citizen science projects. In *Proceedings of the Annual Hawaii International Conference on System Sciences*, pages 610–619.

Regionaal Landschap Kempen en Maasland (2015). Nationaal Park Hoge Kempen. Retrieved from https://www.nationaalparkhogekempen.be.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "Why should I trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Rovero, F., Rathbun, G. B., Perkin, A., Jones, T., Ribble, D. O., Leonard, C., Mwakisoma, R. R., and Doggart, N. (2008). A new species of giant sengi or elephant-shrew (genus Rhynchocyon) highlights the exceptional biodiversity of the Udzungwa Mountains of Tanzania. *Journal of Zoology*, 274(2):126–133.

Rovero, F., Zimmermann, F., Berzi, D., and Meek, P. (2013). "Which camera trap type and how many do I need?" A review of camera features and study designs for a range of wildlife research applications. *Hystrix, the Italian Journal of Mammalogy*, 24(2):148–156.

Rowcliffe, J. M. and Carbone, C. (2008). Surveys using camera traps: are we looking to a brighter future? *Animal Conservation*, 11(3):185–186.

Rowcliffe, J. M., Field, J., Turvey, S. T., and Carbone, C. (2008). Estimating animal density using camera traps without the need for individual recognition. *Journal of Applied Ecology*, 45(4):1228–1236.

Senthilkumaran, N. and Rajesh, R. (2009). Edge detection techniques for image segmentation - a survey of soft computing approaches. *International Journal of Recent Trends in Engineering*, 1(2):250–254.

Shane, J. (2018). Do neural nets dream of electric sheep? Retrieved from http://aiweirdness.com/post/171451900302/do-neural-nets-dream-of-electric-sheep.

Silver, S. C., Ostro, L. E. T., Marsh, L. K., Maffei, L., Noss, A. J., Kelly, M. J., Wallace, R. B., Gómez, H., and Ayala, G. (2004). The use of camera traps for estimating jaguar Panthera onca abundance and density using capture/recapture analysis. *Oryx*, 38(02):148–154.

Simpson, R., Page, K. R., and De Roure, D. (2014). Zooniverse: observing the world's largest citizen science platform. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 1049–1054.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Stanford Vision Lab (2016). ImageNet. Retrieved from http://www.image-net.org.

Steenweg, R. W. (2016). *Large-scale camera trapping and large-carnivore monitoring, occupancy-abundance relationships, and food-webs*. PhD thesis, The University of Montana.

Swann, D. E., Hass, C. C., Dalton, D. C., and Wolf, S. A. (2004). Infrared-triggered cameras for detecting wildlife: An evaluation and review. *Wildlife Society Bulletin*, 32(2):357–365.

Swann, D. E., Kawanishi, K., and Palmer, J. (2011). Evaluating types and features of camera traps in ecological studies: a guide for researchers. In *Camera Traps in Animal Ecology: Methods and Analyses*, chapter 3, pages 27–43. Springer.

Swann, D. E. and Perkins, N. (2014). Camera trapping for animal monitoring: Case studies. In *Camera Trapping: Wildlife Management and Research*, chapter 1, pages 3–12. CSIRO Publishing.

Swanson, A., Kosmala, M., Lintott, C., Simpson, R., Smith, A., and Packer, C. (2015). Snapshot Serengeti, high-frequency annotated camera trap images of 40 mammalian species in an African savanna. *Scientific Data*, 2:150026.

Swinnen, K. R. R., Reijniers, J., Breno, M., and Leirs, H. (2014). A novel method to reduce time investment when processing videos from camera trap studies. *PloS ONE*, 9(6):e98881.

Tobler, M. W., Zúñiga Hartley, A., Carrillo-Percastegui, S. E., and Powell, G. V. (2015). Spatiotemporal hierarchical modelling of species richness and occupancy using camera trap data. *Journal of Applied Ecology*, 52(2):413–421.

Wang, J. and Perez, L. (2017). The effectiveness of data augmentation in image classification using deep learning. Technical report.

Weinstein, B. G. (2015). MotionMeerkat: Integrating motion video detection and ecological monitoring. *Methods in Ecology and Evolution*, 6(3):357–362.

Welbourne, D. J., Claridge, A. W., Paull, D. J., and Lambert, A. (2016). How do passive infrared triggered camera traps operate and why does it matter? Breaking down common misconceptions. *Remote Sensing in Ecology and Conservation*, 2(2):77–83.

Wu, S., Zhong, S., and Liu, Y. (2016). Deep residual learning for image steganalysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Yu, X., Wang, J., Kays, R., Jansen, P. A., Wang, T., and Huang, T. (2013). Automated identification of animal species in camera trap images. *Eurasip Journal on Image and Video Processing*, 2013(52).

Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *Computer Vision - ECCV 2014*, pages 818–833.

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., and Torralba, A. (2016). Learning deep features for discriminative localization. In *2016 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2921–2929.

APPENDIX A

CONFUSION MATRICES

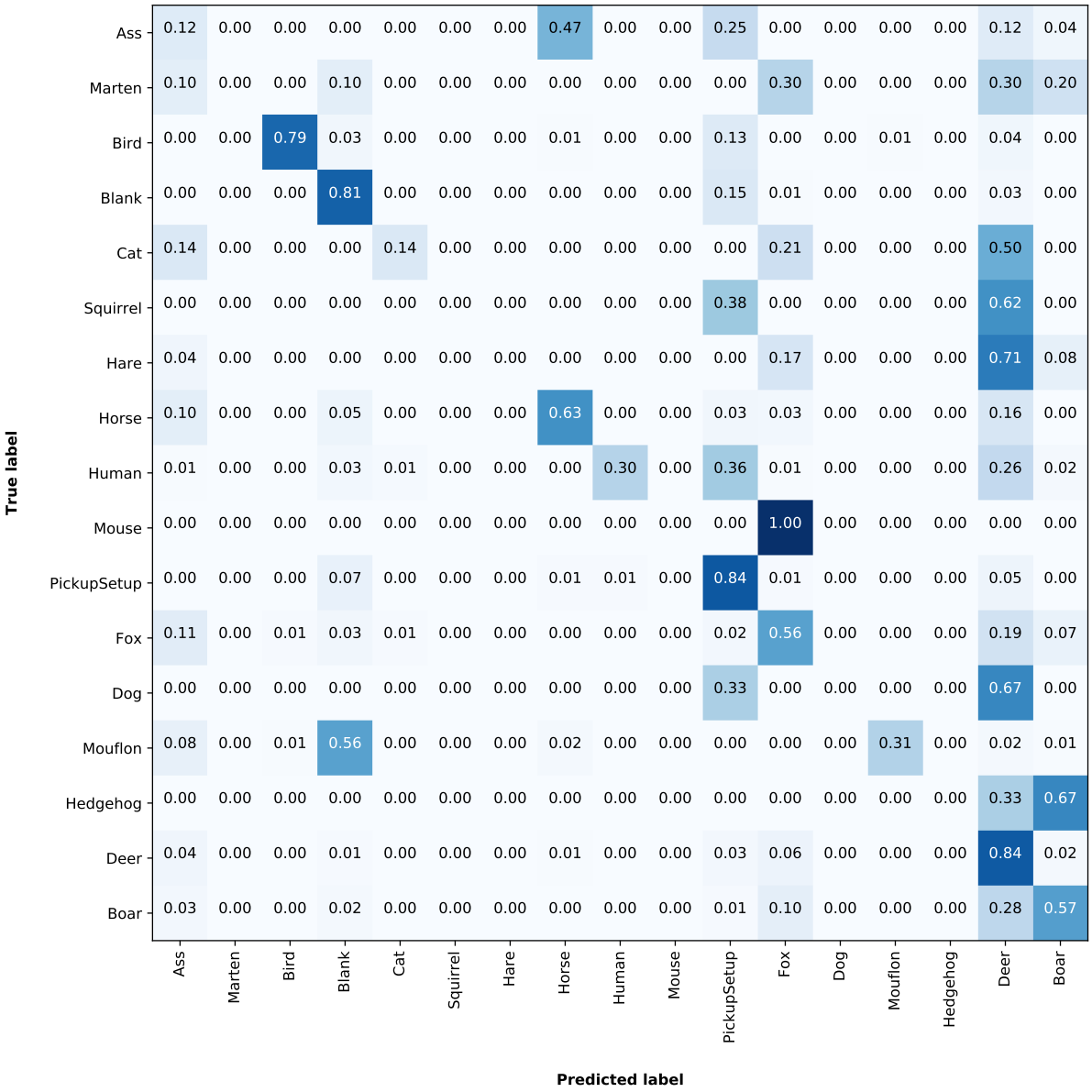
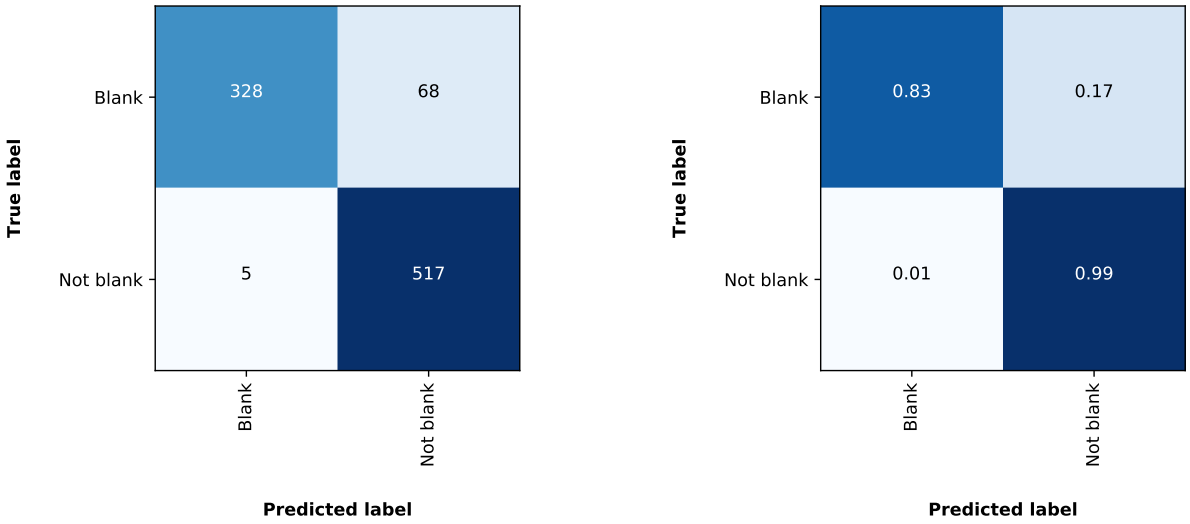
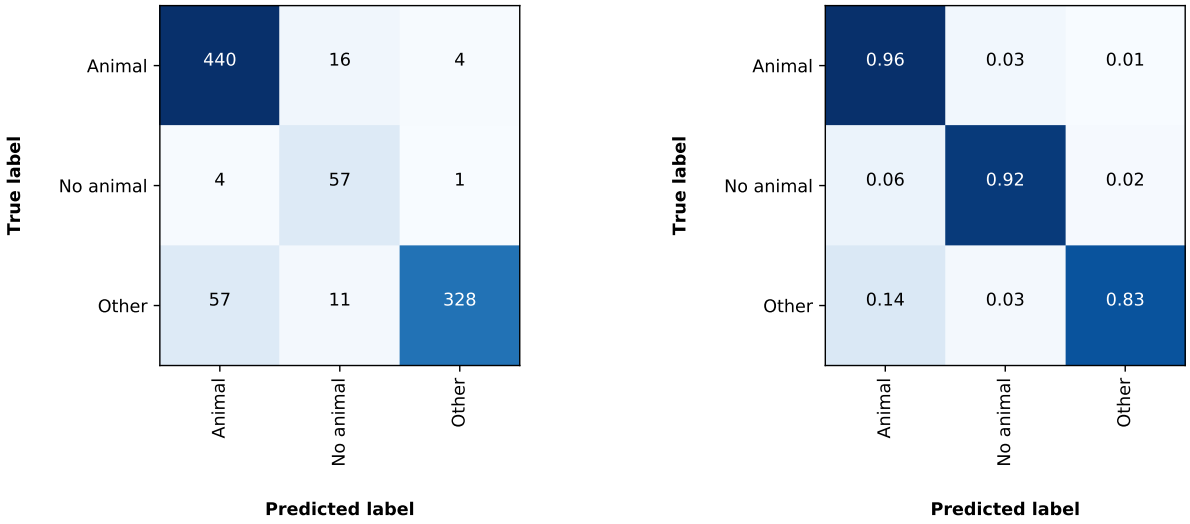


Figure A.1: Normalized confusion matrix when the network is trained without weighting the loss to illustrate the impact of imbalanced data.



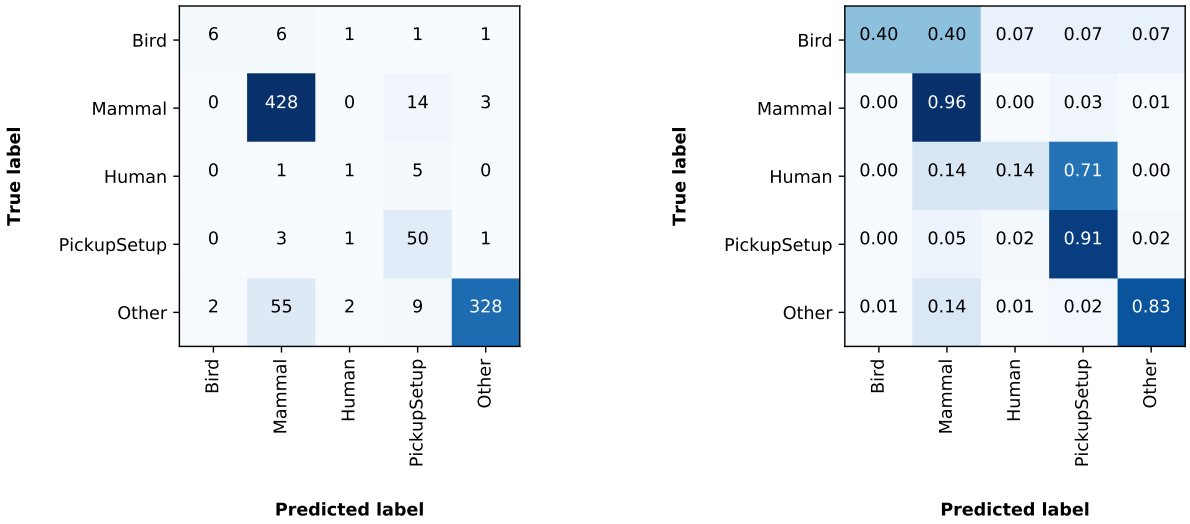
(a) (b)

Figure A.2: Confusion matrix (a) and normalized confusion matrix (b) of the test sequences at the first level of the classification tree.



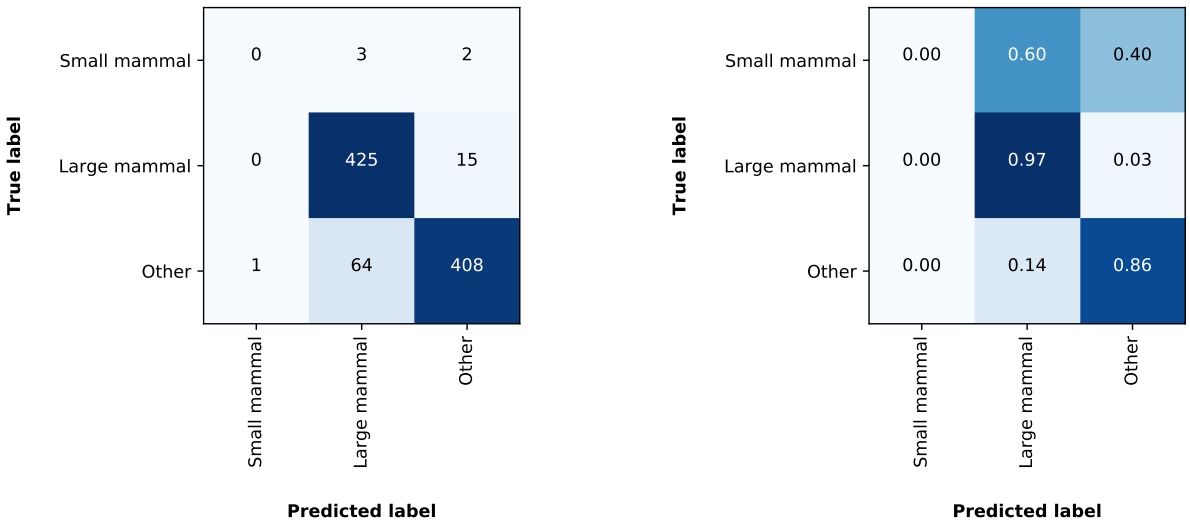
(a) (b)

Figure A.3: Confusion matrix (a) and normalized confusion matrix (b) of the test sequences at the second level of the classification tree.



(a) (b)

Figure A.4: Confusion matrix (a) and normalized confusion matrix (b) of the test sequences at the third level of the classification tree.



(a) (b)

Figure A.5: Confusion matrix (a) and normalized confusion matrix (b) of the test sequences at the fourth level of the classification tree.

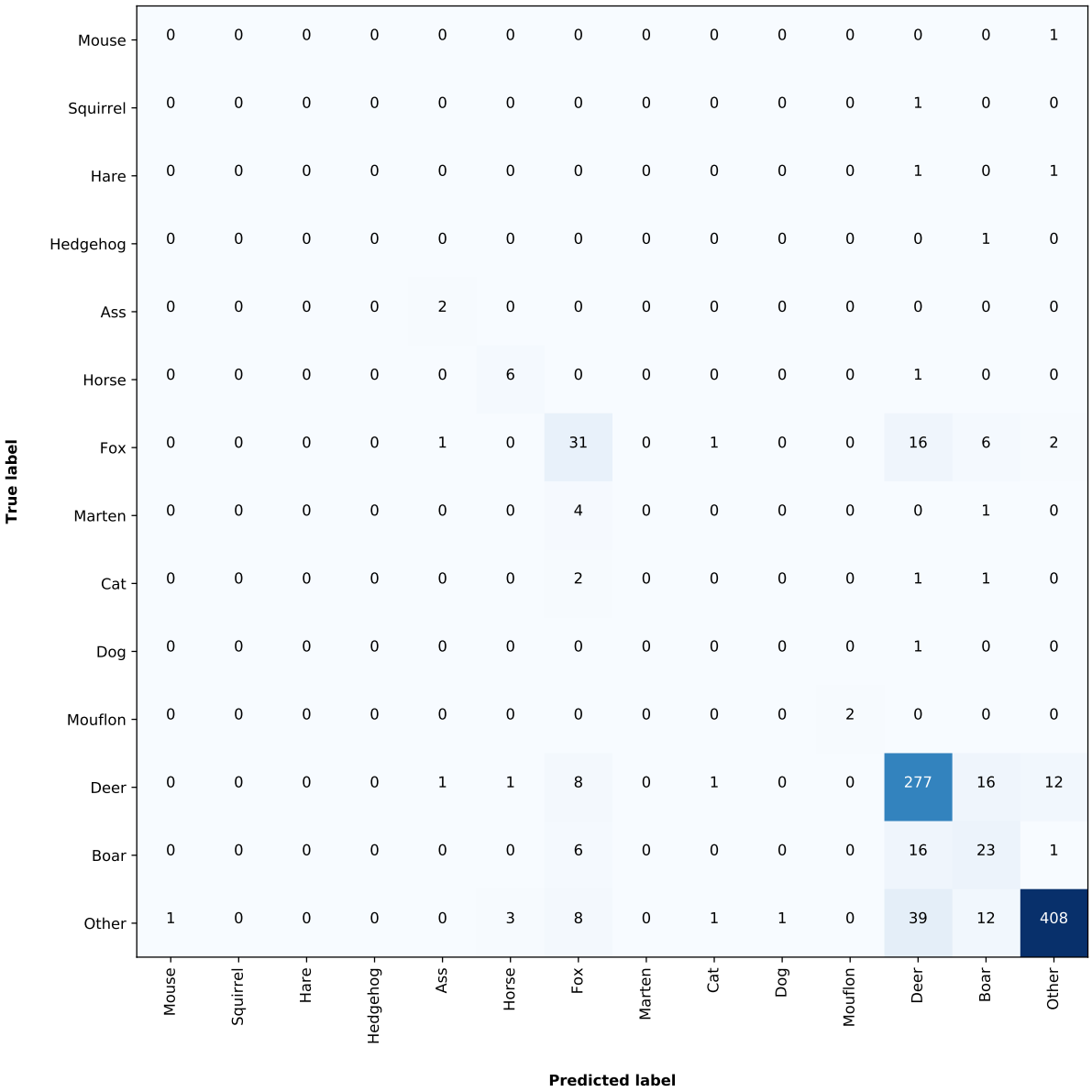


Figure A.6: Confusion matrix of the test sequences at the fifth level of the classification tree.

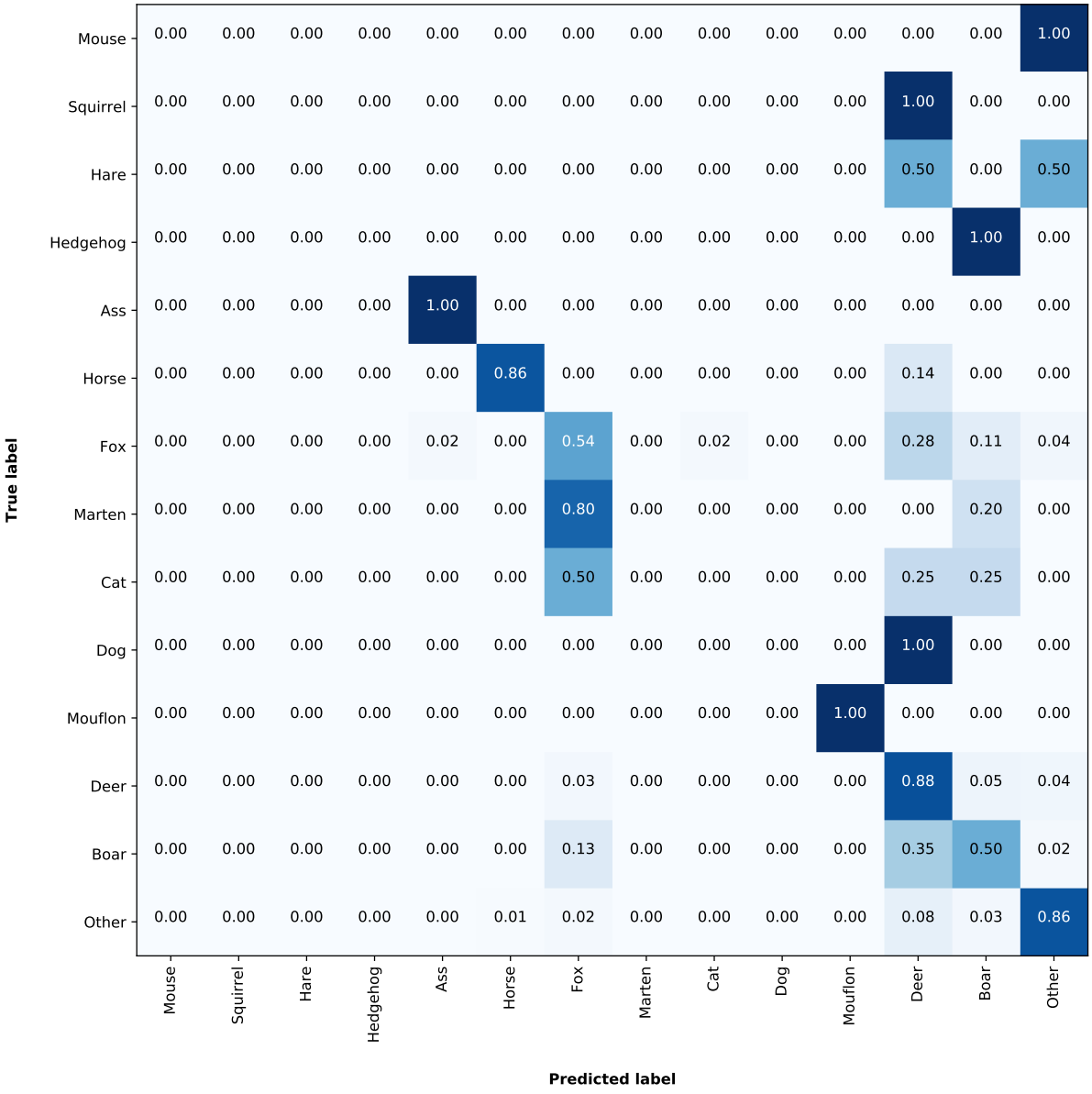


Figure A.7: Normalized confusion matrix of the test sequences at the fifth level of the classification tree.

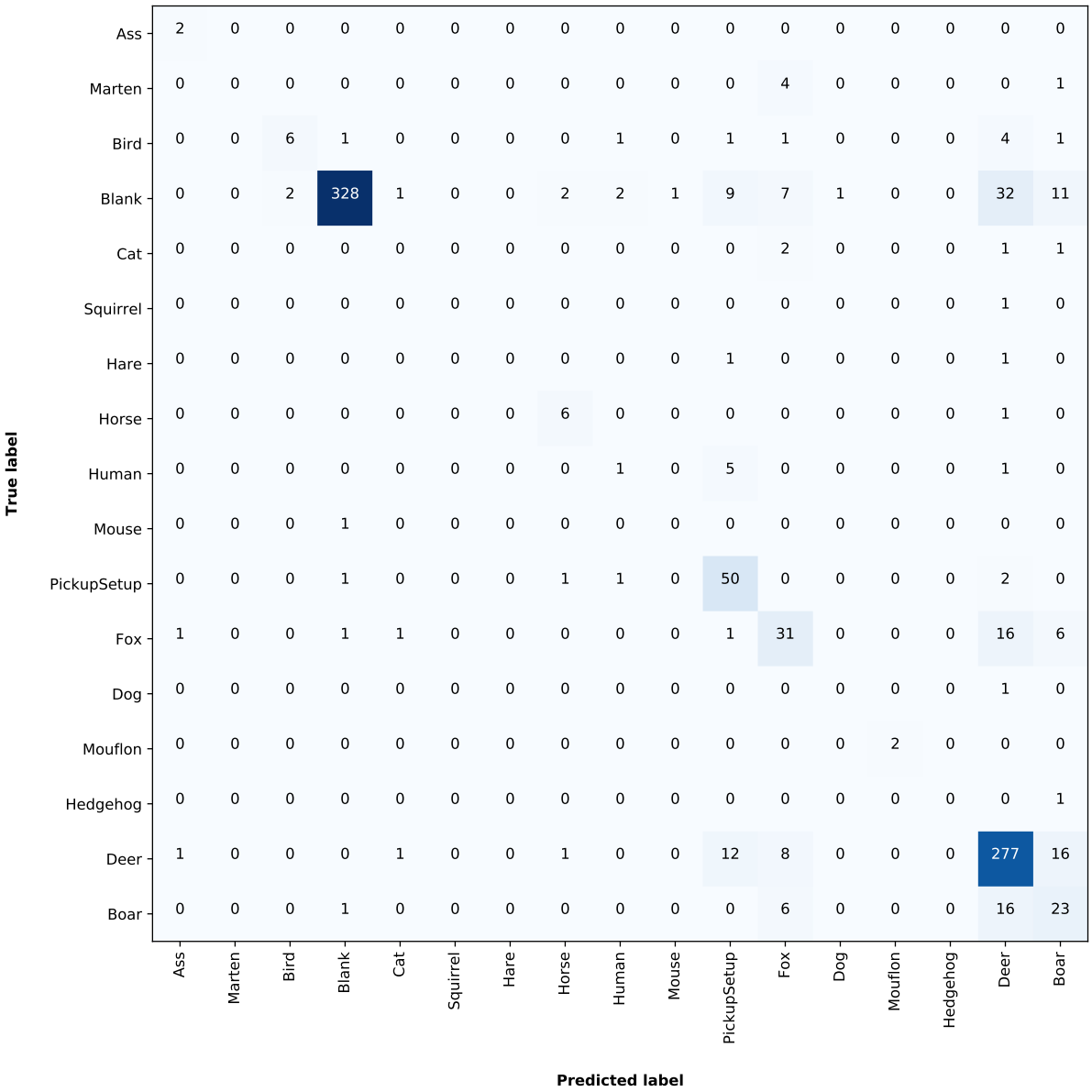


Figure A.8: Confusion matrix of the test sequences with all output classes.

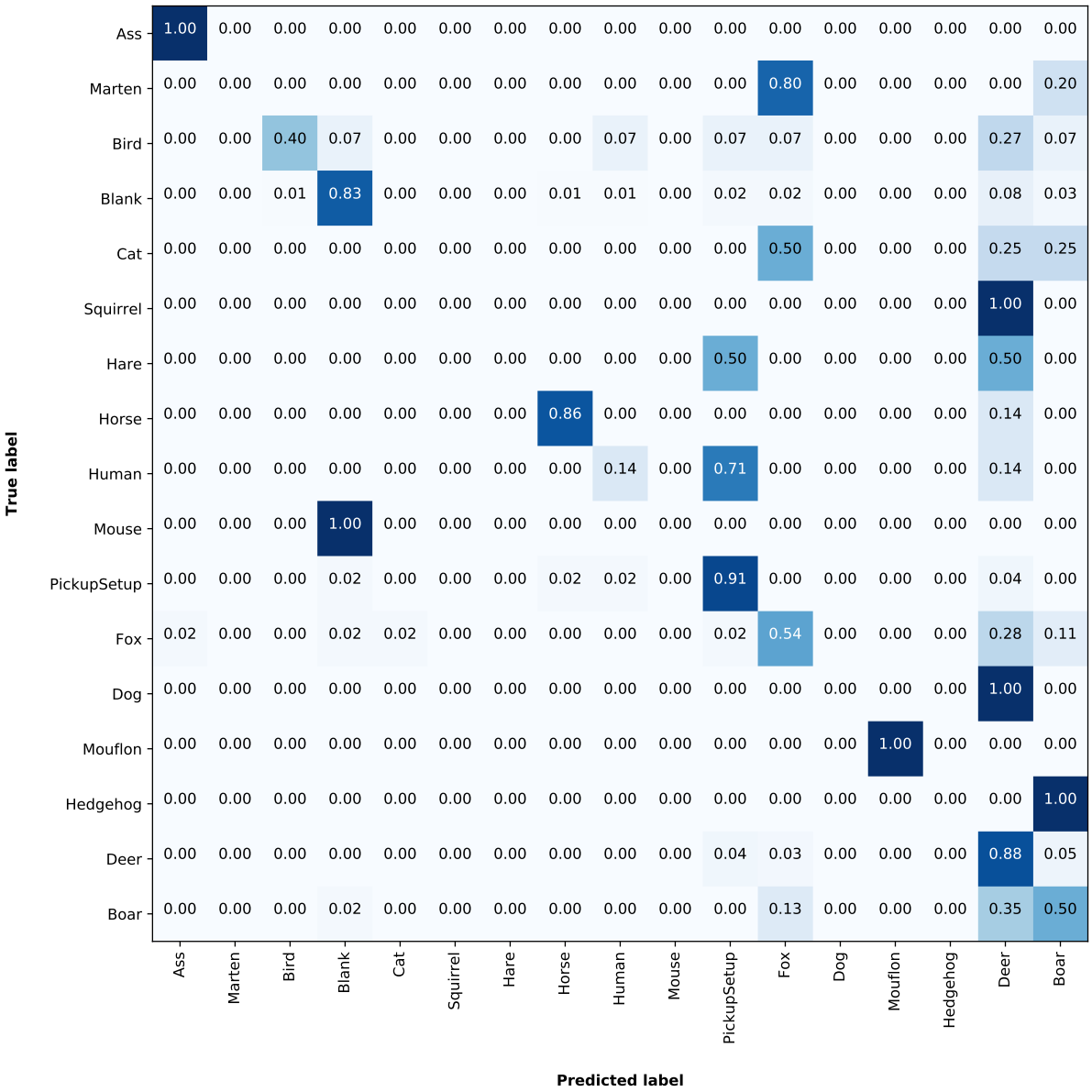


Figure A.9: Normalized confusion matrix of the test sequences with all output classes.