Preface

We would like to express our thanks to a number of persons who helped us to finish this Master's thesis. Our supervisor professor Luc Claesen provided great support during the project with his advice and expertise. Together with our cosupervisor Wout Swinkels, who often offered useful critical thinking and helped refine our work, they also spent a lot of time proofreading this thesis. It is good to thank our external supervisor professor Natalie Beenaerts for the idea of automating the classification of wildlife images. Additionally, we would like to thank Qi Wang for giving us new insights regarding neural networks. We would also like to thank our families, who have shown interest and support throughout the year, as well as our friends who regularly asked how our Master's thesis was coming along. This all helped us complete this project.

- Sven & Laurens

Table of Contents

List of Tables

List of Figures

Nomenclature

AHE	Adaptive Histogram Equalization, page 59
ANN	Artificial Neural Network, page 24
BP	Backpropagation, page 27
CDF	Cumulative Distribution Function, page 83
CLAHE	Contrast-Limited Adaptive Histogram Equalization, page 59
CNN	Convolutional Neural Network, page 28
DL	Deep Learning, page 28
DNN	Deep Neural Network, page 28
FC	Fully Connected (layers), page 31
GPU	Graphics Processing Unit, page 58
HE	Histogram Equalization, page 59
JD	Julian Day, page 40
LR	Learning Rate, page 23
LST	Local Standard Time, page 37
LSTM	Long Short-Term Memory, page 25
RNN	Recurrent Neural Network, page 25
TOA	Top-Of-Atmosphere, page 43
UT	Universal Time, page 40
UTM	Universal Transverse Mercator, page 52

Abstract

Studying biodiversity, which is necessary to understand the human influence on nature, is often done using camera traps. The vast amounts of data recorded this way are labelled manually, which is a labor-intensive task. Automating this process would therefore allow researchers to spend more time examining the results rather than labelling them.

In this research, images are automatically classified using a Convolutional Neural Network. This network is trained on camera trap images from the Snapshot Serengeti project in Tanzania. Additionally, Convolutional Neural Networks trained for regression are used to predict ambient lighting in order to better understand their ability to deduce metadata from an image. Metadata for evaluation is extracted from the image files as well as calculated using empirical models. The inclusion of metadata as additional features to improve classification is examined.

In the conducted experiments, state-of-the-art classification accuracies are obtained. Differentiating between blank images and images containing animals is done with a Top-1 accuracy of 97.33%, while classifying in 48 animal species reaches a Top-1 accuracy of 91.46%. Moreover, it is also shown that, while having no apparent influence in the classification process, metadata can be accurately predicted from images. It therefore can be concluded that using Convolutional Neural Networks for labelling images can significantly speed up biodiversity research.

Abstract in het Nederlands

Biodiversiteitsonderzoek, noodzakelijk om de invloed van de mens op de natuur te begrijpen, wordt vaak uitgeoefend met behulp van cameravallen. Het manueel typeren van de enorme hoeveelheid gegevens die hierbij verkregen wordt, is een tijdrovende activiteit. Bijgevolg zou de automatisatie van dat proces betekenen dat onderzoekers meer tijd zouden hebben om resultaten te bestuderen.

In dit onderzoek worden afbeeldingen automatisch geclassificeerd met behulp van een Convolutioneel Neuraal Netwerk. Dat netwerk leert namelijk afbeeldingen uit het *Snapshot Serengeti* project te herkennen. Bovendien wordt er aan Convolutioneel Neurale Netwerken aangeleerd om door middel van regressie omgevingsbelichting te voorspellen. Zo kan nagegaan worden of de netwerken metagegevens kunnen afleiden uit een afbeelding. Er wordt ten slotte onderzocht of rekening houden met diezelfde metagegevens de classificatienauwkeurigheid verbetert.

De uitgevoerde experimenten behalen hoogstaande nauwkeurigheden. Terwijl blanco afbeeldingen scheiden van afbeeldingen met dieren een Top-1 nauwkeurigheid van 97,33% behaalt, bereikt de classificatie van afbeeldingen in 48 diersoorten een nauwkeurigheid van 91,46%. Daarnaast is aangetoond dat, hoewel ze geen invloed hebben op het classificatieproces, metagegevens nauwkeurig voorspeld kunnen worden vanuit een afbeelding. Er kan besloten worden dat Convolutioneel Neurale Netwerken gebruiken om afbeeldingen te typeren een significante tijdswinst kan betekenen voor biodiversiteitsonderzoek.

Chapter 1

Introduction

Experimentation and observation are two key aspects of science, allowing to verify theorems or come to new insights. In biology, researching biodiversity gives insight in the variability between species and the functioning of ecosystems. In times where climate change and pollution are important topics for society, studying biodiversity would allow to better understand the human influence on nature. In [1, p. 10] (from 2005) is stated that "*changes in biodiversity due to human activities were more rapid in the past 50 years than at any time in human history*". This only adds to the importance of researching biodiversity. One of the practical ways to study biodiversity is studying the species present in an area. Doing this imposes a number of difficulties, however, as it is not self-evident to simply gain all necessary information of one area.

In various national parks, biodiversity is studied using camera traps. Camera traps are an efficient and cost-effective tool in monitoring wildlife. These cameras automatically trigger and capture images of wildlife passing in front of them. The advantage of such cameras is that they are non-intrusive, collect large amounts of data, can access remote locations and require almost no labour. Common research applications include detecting endangered, elusive and new species, and monitoring populations. The main issue faced by researchers is the manual classification of the huge quantity of collected images into useful information, a time-consuming process [2]. Even when applying citizen science (for example as described in [3]) where thousands of volunteers participate in scientific research, in this case image classification, this still is a very large effort. However, with the recent developments in machine learning, specialized algorithms can be developed for automatic image recognition with near-human precision.

In this Master's Thesis, machine learning algorithms are trained and verified on the camera trap dataset provided in the Snapshot Serengeti project [4]. In particular, *Convolutional Neural Networks* (CNN) are trained to classify the images belonging to the different species in the dataset. Two state-of-the-art CNNs are trained: MobileNetV2 and ResNet-50 [5, 6]. These networks are often denoted as *Deep learning* networks due to their deep layered architecture. These CNNs are typically trained using only images, but this research applies additional metadata such as the irradiation of the Sun, the phases of the Moon as well as technical parameters extracted from the cameras. The idea is to give the network a notion of the specific conditions under which the image is captured (e.g. lighting) for better classification performance. All these examinations and experiments contribute to the final goal of this research: Improving and automizing the classification of wildlife camera footage. This way, biodiversity researchers would need to spend less time acquiring data which speeds up their research.

This document is organized as follows. Chapter 2 is the literature study with an in-depth discussion of the Snapshot Serengeti survey, an introduction to machine learning and deep learning, the networks MobileNetV2 and ResNet-50, and the empirical models for solar and lunar parameters. Following this theoretical background is chapter 3, where the implementation of these technologies and models for this problem is discussed. After conducting experiments

using the methods of chapter 3, chapter 4 provides the results and discussion for each experiment. Chapter 5 gives improvements that may be applied in the future, and chapter 6 finally concludes the research.

Chapter 2

Literature study

Before the performed research activity can be discussed, it is necessary to have a firm grasp of the underlying concepts and technology. This chapter will touch on the used dataset, machine learning and the calculation of solar and lunar irradiation, going more in-depth whenever necessary. While it is impossible to describe every detail of for example machine learning, this chapter should provide enough information to understand everything that is discussed and done in the following chapters.

2.1 Snapshot Serengeti

The dataset used in this Master's thesis originates from the Snapshot Serengeti project [4]. Snapshot Serengeti is a survey that has been installing 225 camera traps from June 2010 to May 2013 across a 1,125 km^2 area in the Serengeti National Park in Tanzania (fig. 2.1). The survey's objective is to '*study spatial and temporal dynamics of large predators and their prey*' [4, p. 2]. During the 99,241 camera trap days, a total of 1.2 million image sequences consisting of 1 to 3 images of 48 animal species are captured. To cope with this enormous amount of data, a citizen science website (fig. 2.2) is employed on which the image sets are circulated and labelled by regular users. During classification of an image set, users can specify whether it contains an animal species or is a misfire, the type of species, the number of animals, the behaviour and whether youngs are present. The researchers estimate an accuracy of 96.6% for species identifications and 90% for species counts, and that 75% of the image sets are misfires. The complete dataset is available in [7] as open data. Section 3.1 discusses the dataset more in-depth.

The Snapshot Serengeti survey uses two camera trap models: Scoutguard SG565 with incandescent flash (fig. 2.3) and DLC Covert II with infrared flash. Both camera's are equipped with passive infrared (PIR) sensors that are triggered by a combination of heat and motion, for example by moving animals. Each camera captures 3 images per trigger or capture event during daytime. At night, the Scoutguard camera can only capture 1 image per trigger due to the incandescent flash. The camera sensitivity is set to low to minimize misfires by moving shadows or vegetation. The Snapshot Serengeti team mentions that weather and animals damage 15% of the camera traps annually, and that the camera's batteries and SD cards are replaced every 6 to 8 weeks [4].

Figure 2.1: Geographic location of Serengeti National Park [8, 9].

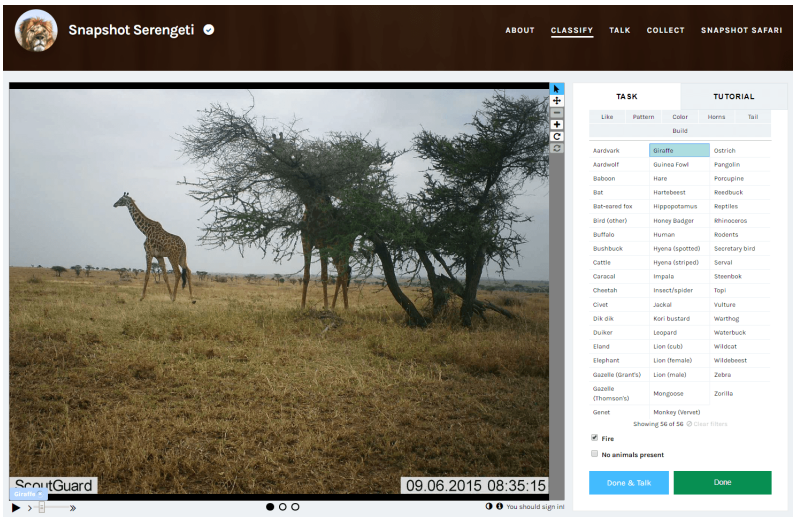


Figure 2.2: Snapshot Serengeti citizen science website [10].

(a) (b)

Figure 2.3: (a) Scoutguard camera trap [11, p. 2]. (b) Placement in Serengeti Park [4, p. 4].

2.2 Machine learning and neural networks

2.2.1 Introduction to machine learning

Despite their popularity nowadays, computing and mathematics are no recent inventions. Already somewhere between 2700 and 2300BC, the ancient Sumerians devised an abacus to perform calculations [12]. During all of history, mathematicians and scientists searched for ways to describe or understand the world. Even algorithms - 'unambiguous specifications of how to solve a class of problems' [13] - were already in use by the Ancient Greeks. An example of this could be the *Sieve of Eratosthenes*, an algorithm to find all prime numbers in a certain range [14, p. 166] [15]. However, although computing and algorithms have been around for a very long time, their use and popularity as known today really can be attributed to the introduction of the computer. Nowadays, countless tasks can be automated using computer algorithms.

There is, however, a discrepancy between non-machine learning algorithms and machine learning algorithms. An algorithm that does not use machine learning usually consists of a number of fixed rules or conditions to process the data. Take for example the sorting of an array of numbers. By following a set number of rules, it is perfectly possible for example to order the numbers from low to high. Another example of non-machine learning algorithms is image processing using carefully chosen, predetermined parameters.
The use of non-machine learning algorithms is not limitless, though, as there is a large amount of situations where they do not provide efficient or quick solutions. One of the classic examples of this is the design of a spam classifier. An intuitive way to do this could be to hard code a very large set of rules to which emails would need to abide in order to not be classified as spam. This solution is naive and inefficient, however, as it is quite impossible to implement. Even if the algorithm would finally work, it could easily be circumvented by slightly adapting the spam emails. Addressing this issue is one of the many potential uses of machine learning algorithms.
The best way to define machine learning is using the definition that is attributed to Arthur L. Samuel, the researcher who coined the term machine learning [16]: Machine learning is the "*field of study that gives computers the ability to learn without being explicitly programmed.*" Another famous definition that puts this a little more concretely is:

> *A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.* [17, p. 2]

That means that instead of hard coding a set of rules or parameters for the algorithm, the algorithm actually has to learn by itself what those parameters ought to be.
Image recognition or *image classification* is one of the major areas where machine learning is utilized. The following sections will discuss the technologies that are used to perform this classification to finally result in an implementation for the specific issue in this project.

2.2.2 Basics of machine learning

For problems in machine learning, there is no baseline algorithm. Both the input and the required output are known. The challenge lies in the transformation from the input to the output [18]. For the example of spam emails mentioned in section 2.2.1, the input data are the received emails and the output data is the classification of those emails in spam and non-spam.

Supervised and Unsupervised Learning

There generally are two kinds of machine learning: *Supervised Learning* and *Unsupervised Learning* [19]. *Supervised Learning* is the kind of learning where a training set is provided, including labels classifying or determining the training examples. That means that for each training example, a label or value is given expressing what output is required for that training example.

Take for example a problem where a machine learning algorithm needs to predict housing prices based on the size of the given houses. In this case, the training set can be written as a matrix $\mathbf{X}$, where the first column contains the house sizes and the second column the corresponding prices:

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ \vdots & \vdots \\ x_{m1} & x_{m2} \end{bmatrix} = \begin{bmatrix} size_1 & price_1 \\ size_2 & price_2 \\ \vdots & \vdots \\ size_m & price_m \end{bmatrix}$$

Figure 2.4 also illustrates this problem. Using a given training set, a learning algorithm is used to form a hypothesis h. This h is the model and takes as input the size of a house to return as output an approximation of the house prize [20]. This way, the algorithm can learn while being supervised: Its results need to be justifiable corresponding to the given results. It can compare each price calculated during training against the given prices and adjust itself accordingly. This way, a '*good and useful approximation*' is found to correctly classify the better part of the data [18, p. 2].
There are different kinds of supervised learning algorithms. Some of the more well known are *regression* and *classification* [20]. Regression will, for a given input, typically give one or more scalar values in a continuous range as output. Classification on the other hand strives to assign the input to one of multiple output classes. Both kinds will be used in this research, but the focus will lie on classification.

For *unsupervised* learning algorithms, however, only input data without labels is given. It is then necessary to find regularities or similarities within this input data [18]. One of the easiest examples of unsupervised learning is *k-means clustering* [21].

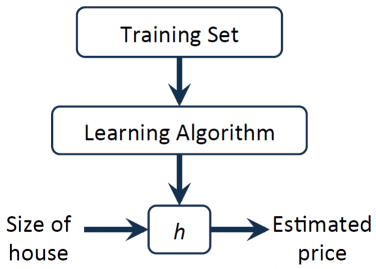


Figure 2.4: Schematic example of the training and use of a hypothesis h to estimate housing prices [20].

Basic elements of machine learning

Even though there are a lot of diverse machine learning algorithms available, a number of basic elements are always returning. As they will be often used throughout this paper, they will succinctly be discussed.
The *dataset* is the departure point for machine learning. By using these data, the necessary algorithm or model has to be automatically generated. While these data can exist in different kinds - as training data or otherwise - it always needs to be present in order to allow machine learning to happen. In this sense, machine learning can also be viewed as a data-driven form of science.
The *model* is the structure that, given a certain kind of input data, will produce the required output data. A good example of such a model is given in 2.2.4, where Artificial Neural Networks are discussed.
The *parameters* are the elements the model is made of. More specifically, the parameters are the

elements that allow for the mathematical interaction with the input data. An easy illustration of this could be equation 2.1 used in linear regression [22]. Let $\mathbf{x}$ be the input data vector and let $\mathbf{W}$ be the parameter matrix. With this information, the output vector $\mathbf{y}$ can be found.

$$y = Wx \tag{2.1}$$

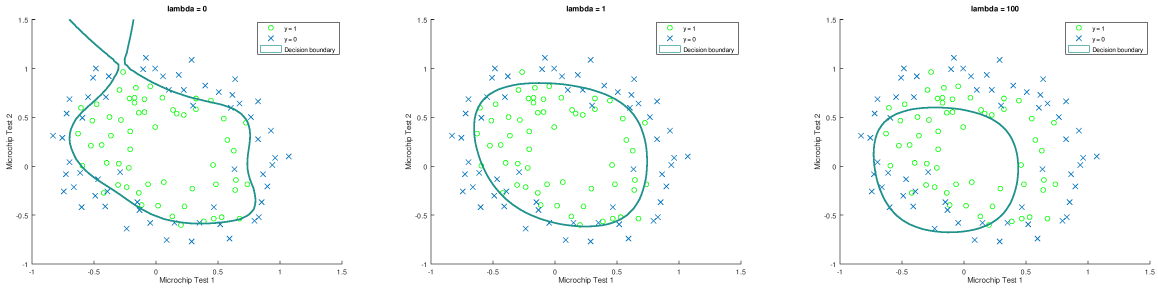
Training, validation and test

For *Supervised learning*, the given dataset ought to be split up in 3 partitions: the *training set*, the *validation set* and the *test set*:

- the training set is used to initially train the model. By calculating the deviation of the output compared to the required output using the current parameters, the model parameters can be adjusted;
- the validation set is used to test the generalization ability [18]. Generalization is further discussed in the next section;
- finally the test set is used to effectively calculate the prediction error for a model. This error can for example be used in official publications or statements. [18].

Overfitting, underfitting and generalization

Whenever machine learning is used, it is important to keep in mind to not overfit (or underfit) the model on the training data. Figure 2.5 based on data from [20] illustrates this very nicely. *Overfitting* the model means that the model will learn too many details concerning the training set, meaning that it will gain high accuracies on the training set. It will however score poorly on new data it has not seen before, as that data most likely does not have the details the model is focused on. *Underfitting* is the opposite of overfitting, meaning that the model is not sufficiently trained to recognize the data. A model that is *generalized* will have a lower accuracy on the training set compared to an overfit model, but it will perform better on data it has not seen before. Regularization then means ensuring that the model can deal with all general appearances of data, rather than focusing on specific details that only a small subset of all possible data contains [18, 20].



(a) Overfitting (b) Generalized (c) Underfitting

Figure 2.5: Based on the regularization parameter λ, different ways to fit the model to the given data are presented. The data represent microchips that are either accepted or rejected after production, based on two tests. The decision boundary visualizes how the machine learning algorithm will decide if a microchip is accepted based on given test results. Depending on λ, the model is either overfitting (a), underfitting (c), or decently generalized (b). Data and figures are introduced by [20].

2.2.3 Loss functions and gradient descent

Loss functions

As previously stated, in all supervised learning problems the given dataset contains the correct output alongside the given inputs. It is therefore necessary to be able to evaluate the calculated output when compared to the required output. This is done using a loss function[1]. In [23], the output of the *supervised learning* system is presented as M(Z^p,W). This stands for the output of the function M for the p$^{\text{th}}$ set of input data Z and the set of parameters W. The cost function then could be E^p=C(D^p,M(Z^p,W)), calculating the deviation of the calculated output of M in comparison to the actual output D for the p$^{\text{th}}$ training example.
Perhaps the most common loss function is the Mean Squared Error MSE, given in eq. 2.2 [20]. In this case, there is a total of m training examples Z^p.

$$MSE = \frac{1}{2m}\sum_{p=1}^{m}(M(Z^p, W) - D^p)^2 \tag{2.2}$$

There exist many more loss functions, but discussing those would be beyond the scope of this research. Whenever a specific loss function is needed, it will be discussed at that point.

Gradient descent

To be able to calculate the error that is made when comparing to the required results is one thing, using this error to update the model to get smaller errors is another. Very often, this is done with *Gradient Descent*. In order to fully grasp what this is, first equation 2.2 should be observed a little more closely. It is easy to see that the loss will be the smallest when the model output M differs very little from the actual output D. As this difference increases, the MSE rises in a quadratic manner. Therefore, if the correctness of output is to be maximized, the loss function needs to be minimized.
Take for example the output function M with two parameters w_0 and w_1 in equation 2.3. The MSE loss function for this output can then be given with equation 2.4.

$$M(Z^p, W) = w_0 z_0 + w_1 z_1 \tag{2.3}$$

$$C(D^p, M(Z^p, W)) = \frac{1}{2m}(w_0 z_0 + w_1 z_1 - D^p)^2 \tag{2.4}$$

Now, if for training example p the required output would be 0, the loss function could be plotted in function of w_0 and w_1. Figure 2.6 shows there is a minimum loss value. At that point, the output value calculated by M using w_0 and w_1 will approximate the required output value the best. This is the point that the model needs to reach.

While this figure provides visual insight in the MSE loss function, it can also help with understanding gradient descent. In gradient descent, the gradient of the loss function is used to calculate how weights need to change in order to decrease the loss. For the given example, the gradient can be found using equation 2.5 [24].

$$\Delta C = \frac{\partial C}{\partial w_0}\Delta w_0 + \frac{\partial C}{\partial w_1}\Delta w_1 \tag{2.5}$$

Equation 2.5 can be interpreted as following: The change in loss is equal to the change of each parameter times the partial derivative of that parameter. In this case, the only parameters are w_0 and w_1. That means that the *gradient* can be written as in equation 2.6, while the parameter vector ΔW is given in equation 2.7 [24].

[1] Other common names are cost function or error function.

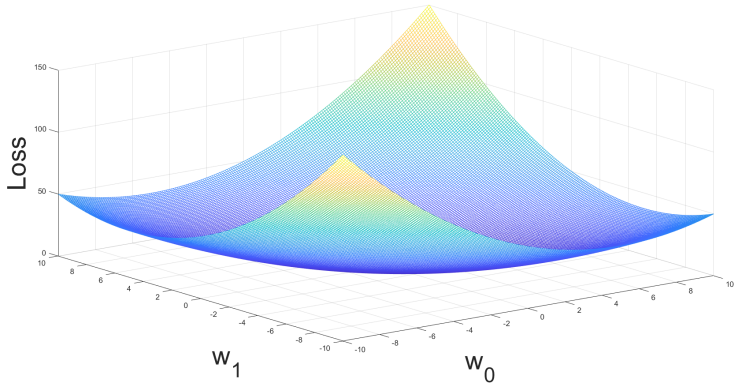


Figure 2.6: Loss function in equation 2.4 with input value Z^p=[1,1] and required output value D^p=0. Notice that there is a minimum where the parameters W will result in the best approximation of the required output. Here w_0 and w_1 respectively are on the x-and y-axes and the loss is on the z-axis.

$$\nabla C = \left(\frac{\partial C}{\partial w_0}, \frac{\partial C}{\partial w_1}\right)^T \tag{2.6}$$

$$\Delta W = (\Delta w_0, \Delta w_1) \tag{2.7}$$

Using these equations, the change in loss can finally be written as a term of the gradient and the parameters (equation 2.8).

$$\Delta C = \nabla C \Delta W \tag{2.8}$$

According to equation 2.8, the change of the parameters ΔW can be chosen in such a way that the change of loss ΔC is negative. Since the loss needs to be minimized, this is exactly what is needed to change the parameters in such a way that they provide better results [24].

Gradient descent finally consists of updating the parameters using the property of equation 2.8 [20, 24]. For the two parameters in the example, this would result in the equations 2.9 and 2.10, where w_0' and w_1' are the updated parameters.

$$w_0' = w_0 - \alpha \frac{\partial C}{\partial w_0} \tag{2.9}$$

$$w_1' = w_1 - \alpha \frac{\partial C}{\partial w_1} \tag{2.10}$$

α is the *learning rate* LR, a coefficient that controls how much the parameters change for a given gradient. With all this information, equation 2.11 finally gives the gradient descent for a parameter vector W, given the gradient of the loss ∇C.

$$W' = W - \alpha \nabla C \tag{2.11}$$

The entire process of gradient descent can intuitively be understood using figure 2.6 again. If the current loss were a person standing on a point of the graph, gradient descent would be that person stepping downwards to the bottom of the graph. Every step he takes would be in the direction of the steepest slope. The learning rate would in that case be the step size. While in reality no loss functions are as simple as in this example, it does give a good insight in how gradient descent can be used to train a network.

2.2.4 Artificial Neural Networks

One of the most well known kinds of machine learning methods is the Artificial Neural Network (ANN). This section discusses neural networks, their construction and their different components.

Neurons

Figure 2.7 gives the basic representation of the ANN. Each neuron is connected to other neurons by its inputs, its outputs or both. Neurons that only provide output signals are the neurons of the *input layer*, while neurons that only have inputs belong to the *output layer*. All other neurons have both input and output signals and belong to a *hidden layer* [20]. Each signal also has a weight associated with it. The simplest mathematical model of a neuron is given in equation 2.12, according to [18].

$$y = \sum_{j=1}^{d} w_j x_j + w_0 \tag{2.12}$$

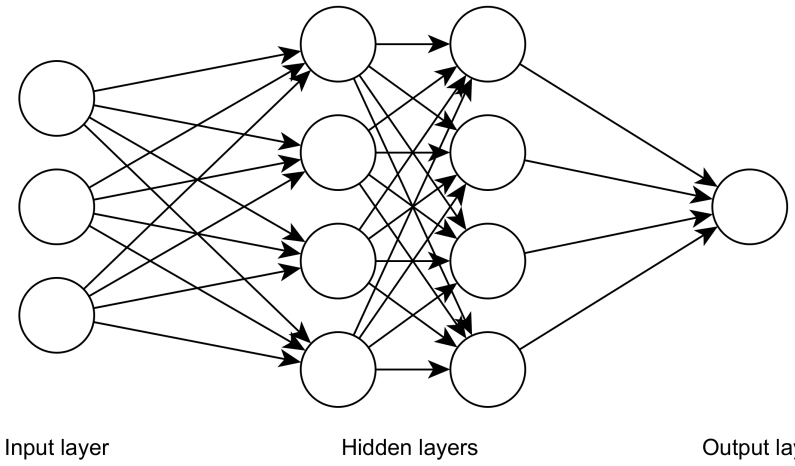


Figure 2.7: Basic representation of a neural network. Each circle is a neuron, while each arrow signifies a connection between two neurons. Note: the *bias unit* is not included in this schematic.

Let the neuron have n input signals x_j with weights w_j. In that case w_0 is the weight belonging to the *bias unit* x_0 with a constant value of 1. Figure 2.8 visualizes this model. An equation can also be given for weights and input values that are stored in matrices. Let $\boldsymbol{w}$ be the 1×n+1 matrix containing all weights, and let $\boldsymbol{x}$ be the n+1-dimensional vector containing all input signals:

$$\mathbf{w} = \begin{bmatrix} w_0 & w_1 & \dots & w_n \end{bmatrix}$$

$$\mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$$

If this is the case, equation 2.13 gives a vectorized version of equation 2.12, allowing for faster computation [18, 20].

$$y = \mathbf{w}^T \mathbf{x} \tag{2.13}$$

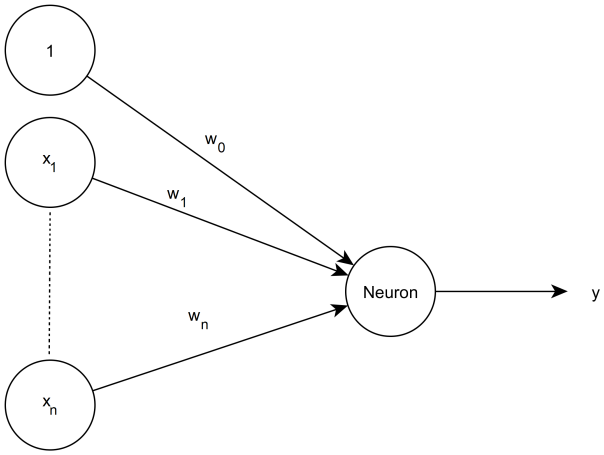


Figure 2.8: A schematic drawing of a neuron is depicted here. Taking all of its inputs x_j and the corresponding weights w_j, the neuron calculates an output y.

Activation function

In the previous section, an example of a neuron with a simple linear activation function is given. While it makes for an easy example, there is a litte more to the activation function. The activation function is used to transform the given inputs into an output [25]. It also allows for a non-linearity to be implemented. Non-linearities allow the network to learn relations between input and output that are non-linear and more complex than linear relations [26]. [27] proves that a neural network with only one hidden layer with non-linear sigmoidal activation functions can approximate about any decision region. On the contrary, in [28] is stated that a multilayer neural network without non-linearities could be perfectly recreated using a single layer neural network (with sigmoidal non-linearities). There are many kinds of activation functions. This section will discuss the most important and common ones.

The sigmoid function is an older non-linear activation function, traditionally used in feed-forward neural networks. It is currently still used in Recurrent Neural Networks (RNN) [29], for example in Long Short-Term Memory (LSTM) Networks[2]. Equation 2.14 presents this sigmoid function [31], a graphical representation is given in figure 2.9.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.14}$$

Typical for this activation function (and all other commonly used activation functions) is that it has an output value for all real input values. In addition, it has a positive derivative for any input value [32]. It is important to note that the sigmoid function has a number of disadvantages however. These disadvantages include [31]:

- gradient saturation,
- slow convergence,
- nonzero centered output,
- sharp damp gradients.

The ReLU [33] nonlinear activation has demonstrated to be preferable in deep feedforward neural networks (due to e.g. the vanishing gradient problem in sigmoid that leads to very slow convergence in deep neural networks). It is first introduced by [34] and provides better generalization and performance in deep learning when compared to the sigmoid activation function [31].

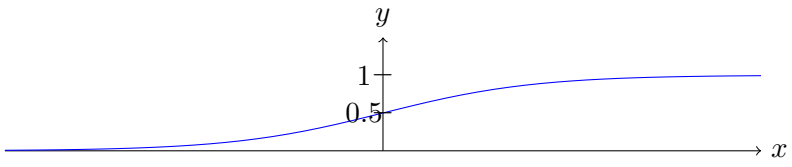


Figure 2.9: The sigmoid activation function.

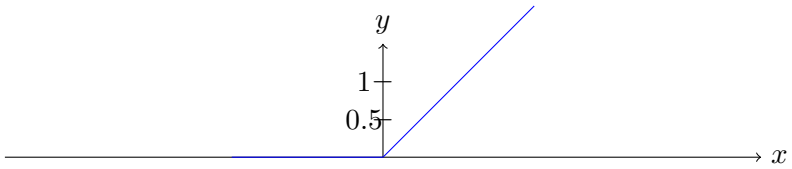


Figure 2.10: The rectified linear unit activation function.

As also shown in equation 2.15 and figure 2.10, the ReLU is significantly simpler than its sigmoid counterpart. It can also be seen as a threshold operation [31].

$$f(x) = max(0, x) = \begin{cases} x, x \geq 0 \\ 0, x < 0 \end{cases} \tag{2.15}$$

While this activation function is not differentiable in zero, this is no issue as a value of exactly zero will hardly ever be used as input value. The left and right derivates still exist for zero, which means that while they are not equal, any value extremely close to zero will still result in a valid derivate [35]. The reason the sigmoid function (among others) suffers from the vanishing gradient problem is that values much greater or smaller than zero result in a derivate that is nearly zero, which significantly reduces propagation throughout the network [36]. ReLu does not have this issue as values far greater than zero still have a relevant derivate, while values smaller than zero always result in a derivate of zero. In [37], it is stated that the use of ReLU results in easier optimization, faster convergence, better generalization and faster computation.

The network

While the study of the individual neurons and their functioning is important, the real strength of the artificial neural network lies in the combination of many neurons in layers. As previously discussed, an ANN contains an input layer, an output layer and at least one hidden layer [20]. Figure 2.7 shows this concept for an ANN with two hidden layers. Concretely, the hidden layers map the input layer to the output layer.
The number of outputs for a neural network depends on the task at hand. If, for example, a network needs to decide whether a received email is spam or not, the task can be done using only one output. Providing different elements of the email as input, the network calculates whether the email is spam or not and output a number between 0 and 1. In this case, a zero could be the equivalent of *certainly not spam*, while the 1 would be the equivalent of *certainly spam* [20]. On the contrary, if the network needs to recognize for example certain animal species on an image, multiple outputs are necessary. Each of the outputs then corresponds to one of the possible classes the image can belong to. Take for example a network that is trained to recognize farm animals. Possible output classes are then: *Cow*, *Sheep*, *Chicken* and *Dog*. In this ANN, every input image will result in a score for each output class. A simple way to decide on the animal portrayed on the image could then simply be to take the output class with the highest value.
Whenever using ANNs or derivatives of that technology, *forward propagation* and *backpropagation* are two very important concepts [20]. They will be briefly discussed in the following paragraphs.

[2] An in-depth introduction to RNN and LSTM can be found in [30].

Forward propagation

Forward propagation is the process in which the input values are forwarded throughout the ANN to finally result in some output value. To illustrate this, an example based on the course example in [20] will be used. Considering figure 2.7, let $\boldsymbol{x}$ be the 4-input vector containing the *bias unit* 1 and the 3 input values. Furthermore, let $\boldsymbol{w^{(k)}}$ be the matrix containing the weights $w_{\mathrm{ij}}^{(\mathrm{k})}$ for the layer k, with each row i corresponding to the weights for the inputs of the i-th neuron of that layer. The first 4×4-matrix $w^{(1)}$, containing the weights mapping the input values into the first hidden layer, is given below:

$$\boldsymbol{w}^{(1)} = \begin{bmatrix} w_{00}^{(1)} & w_{01}^{(1)} & w_{02}^{(1)} & w_{03}^{(1)} \\ w_{10}^{(1)} & w_{11}^{(1)} & w_{12}^{(1)} & w_{13}^{(1)} \\ w_{20}^{(1)} & w_{21}^{(1)} & w_{22}^{(1)} & w_{23}^{(1)} \\ w_{30}^{(1)} & w_{31}^{(1)} & w_{32}^{(1)} & w_{33}^{(1)} \end{bmatrix}$$

In addition, let *g(x)* be the activation function of an individual neuron. Moreover, let $a_i^{(\mathrm{k})}$ be the output of a neuron i on layer k. The vector of the outputs of all neurons on layer 1, $\boldsymbol{a^{(1)}}$ can then be given:

$$\boldsymbol{a}^{(1)} = \begin{bmatrix} g(w_{00}^{(1)}x_0 + w_{01}^{(1)}x_1 + w_{02}^{(1)}x_2 + w_{03}^{(1)}x_3) \\ g(w_{10}^{(1)}x_0 + w_{11}^{(1)}x_1 + w_{12}^{(1)}x_2 + w_{13}^{(1)}x_3) \\ g(w_{20}^{(1)}x_0 + w_{21}^{(1)} + x_1 w_{22}^{(1)}x_2 + w_{23}^{(1)}x_3) \\ g(w_{30}^{(1)}x_0 + w_{31}^{(1)}x_1 + w_{32}^{(1)}x_2 + w_{33}^{(1)}x_3) \end{bmatrix}$$

Here x_0 is the value 1 of the *bias unit*. Considering all this information, equation 2.16 shows that there is a more succinct notation for finding $\boldsymbol{a^{(1)}}$.

$$\boldsymbol{a^{(1)}} = g(\boldsymbol{w^{(1)}x}) \tag{2.16}$$

Equation 2.17 then gives the more general notation for finding the output of a layer k.

$$\boldsymbol{a^{(k)}} = g(\boldsymbol{w^{(k)}a^{(k-1)}}) \tag{2.17}$$

Finally, for the given example, the output is given in equation 2.18.

$$y = g(\boldsymbol{w^{(3)}a^{(2)}}) = g(\boldsymbol{w^{(3)}}g(\boldsymbol{w^{(2)}a^{(1)}})) = g(\boldsymbol{w^{(3)}}g(\boldsymbol{w^{(2)}}g(\boldsymbol{w^{(1)}x}))) \tag{2.18}$$

This simple example shows how the output of an ANN can be calculated using the weights and the input values. Note that the bias units of the hidden layers have been left out for simplicity's sake.

Backpropagation

In machine learning, a training process is necessary to tune all the parameters to correctly calculate the output values given some input values. For an ANN, these parameters are the weights connecting the neurons. In order to train these parameters, a backpropagation algorithm BP is used. The principle of BP has first been discussed in [38], originally published as a master's thesis [19]. One of the main papers introducing BP for an actual ANN can be attributed to [39], where a theoretical base for BP in ANNs is given. In its essence, BP can be viewed as an operation very similar to forward propagation. As previously mentioned, in forward propagation the input data is pushed through the network to calculate the output. In BP however, the output value is used to adjust the weights of the network. This is done by calculating the error of the output during training (compared to the required output for the given input) and then

propagating that error from the back to the front of the network. The main problem in doing this is finding how much a single parameter influences the final output [39].
As previously mentioned, gradient descent is used to adjust parameters to correctly process data. In a simple linear machine learning model, this process is fairly straightforward. In ANNs however, backpropagation is necessary to perform the gradient descent correctly. Gradient descent works by calculating the partial derivatives of the loss with regards to each parameter for an input training set [23]. These partial derivatives can then be used as the gradient to calculate the update values for those parameters. Explaining this principle in-depth would lead too far, however. A good explanation of BP can be found in [23], among other sources.

2.2.5 Deep learning

Conventionally, two kinds of learning with ANNs exist. *Shallow* ANNs are built using only a few hidden layers. *Deep* ANNs on the other hand, consists of many hidden (or other) layers. Machine learning using Deep Neural Networks DNN is called *Deep Learning* DL. While many types of DNNs exist, the focus in this research will lie on the Convolutional Neural Networks. More information concerning DL and its history can be found in [19].

2.3 Discussion of convolutional neural networks

2.3.1 Convolutional Neural Networks

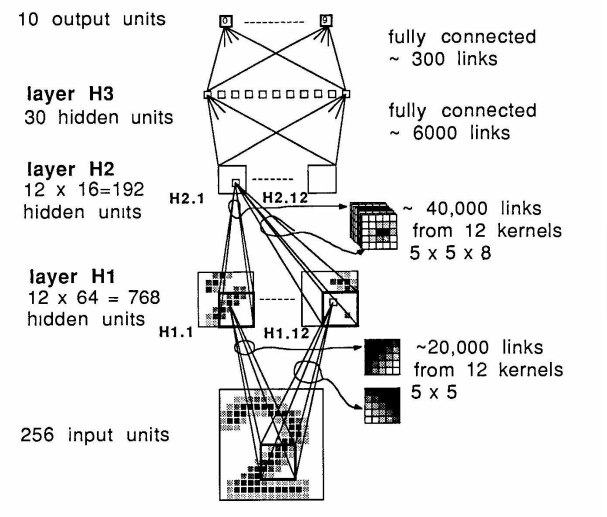
As a specialized kind of ANNs, Convolutional Neural Networks (CNNs) are a technology used for image recognition (among other things). One of the earliest examples of something resembling a CNN is [40, p. 548], where small details or features in a 16×16 image are used to find information about general features in that image (see also figure 2.11a). The author, Yann LeCun, further develops this network into LeNet-5 [41, p. 7], where the characteristic shape of the CNN can already be clearly seen (figure 2.11b). At this point the name *Convolutional Network* also is in use. More than a decade later, another CNN revolutionized image recognition. AlexNet [42] convincingly wins the ImageNet Large Scale Visual Recognition Challenge ILSVRC [43, 44], gaining significantly higher accuracy ratings than the competition. In its essence, AlexNet does not structurally differ much from LeNet-5, sporting similar layers as can be seen in figure 2.12. An important evolution however is that AlexNet can rely on much more powerful hardware to train, as CNNs were too expensive to handle before that time, as well as the use of the ReLU (Rectifying Linear Unit) as a non-linear activation instead of the traditional sigmoid function. Following AlexNet, CNNs rapidly grew in popularity while delivering increasing accuracies in for example VGGNet [45], GoogLeNet (Inception.v1) [46,47] and ResNet [6]. The following sections explain the key components of a CNN, allowing for a better understanding of the functioning of a CNN.

2.3.2 Convolution Layers

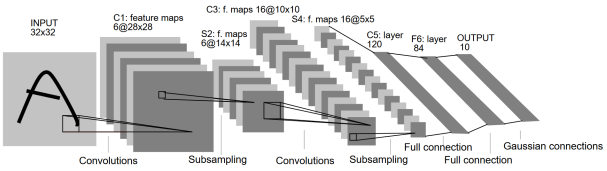
The namesake layer of the CNN is the convolution layer. This is the layer responsible for recognizing distinct features within an image, which is necessary for classification.

Convolution for digital images

Before the convolution layer in the CNN can be discussed, it is important to define what convolution is within the context of image processing. Let f be a grayscale image. This image can then be represented as a $m \times n$ matrix, where m and n respectively are the number of pixel columns and rows of the image. Additionally, let w be a $(2k+1) \times (2l+1)$ window. Considering m, n, k and $l \in \mathbb{N}$, f and w can be defined using equations 2.19 and 2.20, for the discrete variables x and y. The convolution of the image f by window w can then be written following



(a) Digit Recognition



(b) LeNet-5

Figure 2.11: Two early CNNs designed by Yann LeCun [40, p. 548] and [41, p. 7].

equation 2.21 [48].

$$f(x,y) = \begin{cases} f(x,y) & \text{if } 0 \leq x < m \text{ and } 0 \leq y < n \\ 0 & \text{otherwise} \end{cases} \tag{2.19}$$

$$w(x,y) = \begin{cases} w(x,y) & \text{if } -k \leq x \leq k \text{ and } -l \leq y \leq l \\ 0 & \text{otherwise} \end{cases} \tag{2.20}$$

$$(f * w)(x,y) = \sum_{s=-k}^{k} \sum_{t=-l}^{l} w(s,t) f(x-s, y-t) \tag{2.21}$$

Essentially, w moves over f while continuously calculating the dot product of itself and the pixels of f it is currently aligned with. The result is then the image formed using these dot products. Taking the convolution of a RGB image and w can be done by convolving every color component of the image with w seperately, and then concatenating the results. Alternatively, a $(2k+1) \times (2l+1) \times 3$ window can be used to convolve with the image and retain a result $a \times b \times 1$, where a and b are the dimensions of the image after convolution. This principle is demonstrated in figure 2.13.

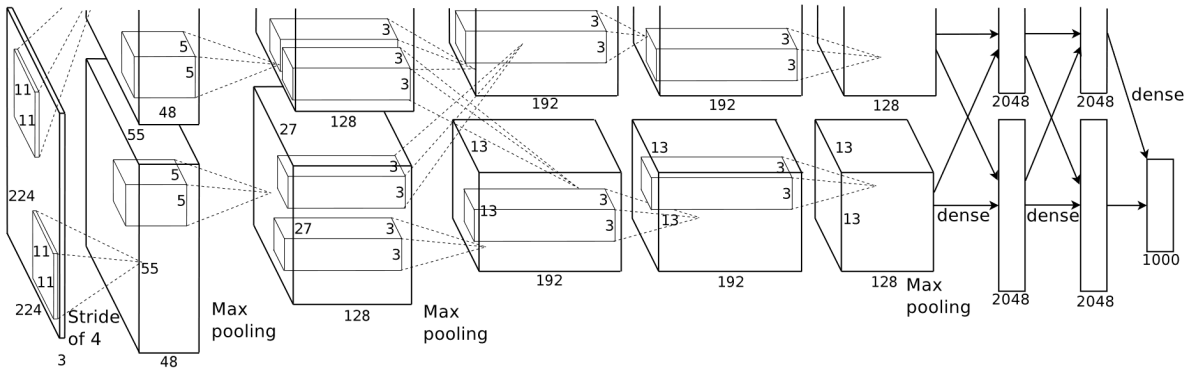


Figure 2.12: AlexNet, as introduced in [42, p. 5].

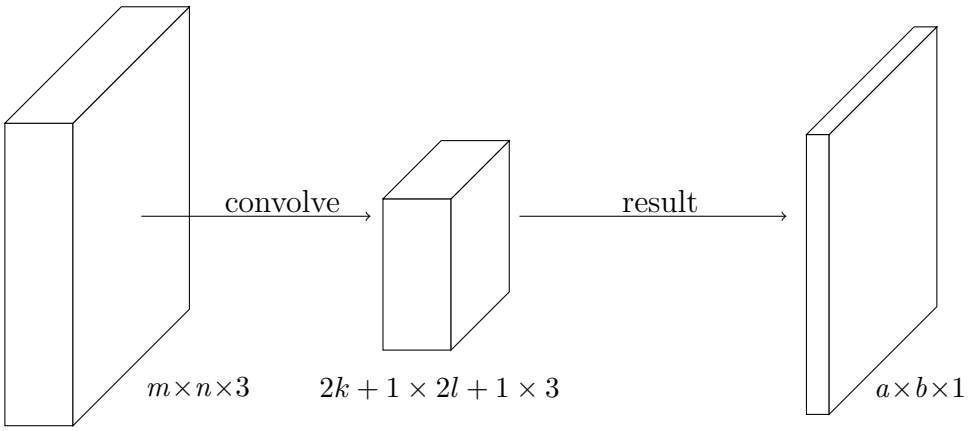


Figure 2.13: The functioning of the convolution layer is demonstrated here.

Additionally, a convolution operation can have a stride. The stride is the number of steps s after calculating a dot product. For the convolution in equation 2.21, the stride is 1. It is important to note that not all convolutions seem possible. Take for example the convolution of a $4 \times 4 \times 3$ image with a $3 \times 3 \times 3$ window with a stride of 2. It is easy to see that the window will, after one step, exceed the image. To resolve this issue, padding can be used. Various kinds of padding exist, such as for example zero-padding. Zero-padding is the operation artificially adding zeros to the border of the image to enable specific convolution operations. In the given example, the zero-padded border needs to be one zero thick to allow the convolution to take place. In equations 2.19 and 2.20 this zero-padding is included by having the value of respectively f and w be zero outside of the domain.
The use of this kind of convolution is prevalent in digital image processing, where specific values for w are used to detect or extract certain features in digital images [49]. If the parameters of w (the values of w) are trained using machine learning, w can be trained to detect certain details or features in an image.

Convolution in convolution layers

Using this prelimenary definition of convolution, the convolution layers can be discussed. This section is based on [22], where a more in-depth explanation can be found. As demonstrated for a single convolution operation in figure 2.13, the convolution layer takes an input image and performs a convolution operation upon it. Typically, the convolution has specific 3D dimensions and a certain stride. The result then has a reduced height and width (if no excessive zero-padding is used), and a depth of 1. [22] provides more insight regarding the exact change in height and width.
Fundamentally, the convolution layer is a stack of u windows that are all applied to the same input image (or activation maps). The resulting stack of output images (activation maps[3]) then has a depth of u, given that the result of each convolution has a depth of 1. Specifically, that means that the chosen number of windows determines the depth of the result. The trainable parameters of these layers then are the values of each individual window. Applying machine learning allows for each convolution window to be trained to recognize certain features. Typically, the first convolution layers detect general features (colors, borders, textures, larger objects,...), while the final convolution layers help classify into the output classes (as layers are deeper into the network, the detected features grow more specific) [22].

[3]Essentially, an activation map can be seen as the output image after the convolution of an input image. Activation maps are usually not regarded as images, however, as they are inside the CNN and have a depth that is far larger than the usual depth of an image (RGB images have a depth of 3). Where images are meant to be visualized, activation maps are intermediate values in a large calculation not meant to be visualized.

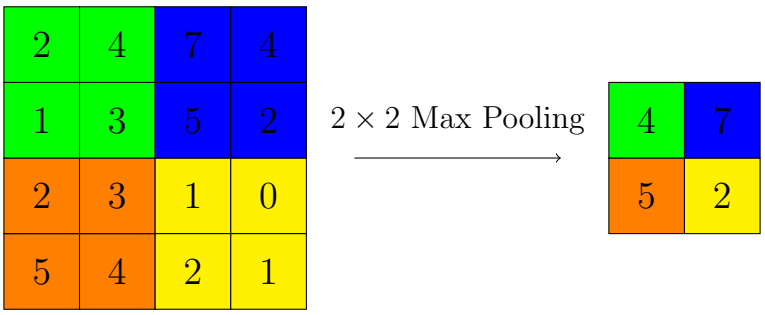


Figure 2.14: Max Pooling according to [22]. The image is divided into 2×2 sections, the maximum of each section is taken and finally used to get the values of the downsampled image.

2.3.3 Pooling Layers

Pooling Layers fulfill a downsampling role in the CNN [22]. They essentially compress the height and width of one image or activation map. This is done by taking a section of the input, performing an operation on that section to finally return the result in a single pixel or scalar. Note that this operation is performed on each activation map in a stack with depth u. Two of most commonly used layers are max pooling (performing a maximum operation) and average pooling (performing an averaging operation). Figure 2.14 provides a simple example of max pooling, inspired by [22].

2.3.4 Fully Connected Layers

Fully connected FC layers finally tie the CNN together by performing classification. They take the data presented by the convolution and max pooling layers and process it the way a regular ANN would. This is done by performing a pooling operation on the final convolution activation maps and flattening the result into one feature vector or hidden layer. Figure 2.15 demonstrates this concept. Note that many modern networks (for example ResNet [6]) use global average pooling (section 2.3.5) rather than regular pooling, which alleviates the need for flattening. The output of this single layer (with an arbitrary number of additional FCs) is then finally fed into a final layer with a *softmax* activation.
Introduced in [50], the softmax function takes a number of input values and scales them between 0 and 1. The general formula for this function is given in equation 2.22 [50, p. 231]. If N would be the number input values I_j, O_j would be the corresponding output values according to the softmax function.

$$O_j = \frac{e^{I_j}}{\sum_{k}^{N} e^{I_k}} \tag{2.22}$$

Taking the output from the previous FC and bundling it for N separate classes (hidden units), all inputs for the softmax function can be found. A score is then calculated for each class using equation 2.22 to achieve a prediction between 0 and 1 (where the sum of all scores is 1). These values are the certainty of the network that the corresponding classes are present on the input image.
Overall, this means that first the convolution layers detect the presence of particular features in an image. Next, the FC's task is to take all this information and tie it together to classify the image. If for example the image contains wheels, headlights and side mirrors, the FCs could give the class *car* a high classification score.

2.3.5 Global average pooling

Where standard pooling layers perform downsampling on an input image, global average pooling reduces the entire input activation map into a single value. First introduced in [51], its authors

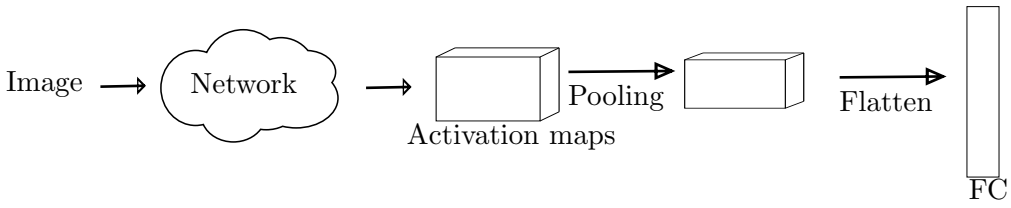


Figure 2.15: General idea of turning activation maps into a FC.

argue that fully connected layers (see section 2.3.4) are prone to overfitting. To prevent this, they reduce the need of fully connected layers by directly feeding the pooled output of the activation maps into the softmax FC layer. This is done by calculating the average value of each activation map rather than using max/average pooling and retaining a smaller activation map. Intuitively, this means that each activation map is responsible for the confidence score of one particular category. Taking the example of the car in section 2.3.4, one activation map should be responsible for detecting the presence of a headlight. Another map should then give a confidence score for the presence of side mirrors, and a third should detect wheels. In [51], the global average pooling is presented as an alternative to other kinds of regularization for the FC. Networks, such as ResNet [6], have adopted this practice. Figure 2.16 gives a simple representation of the principle of global average pooling.

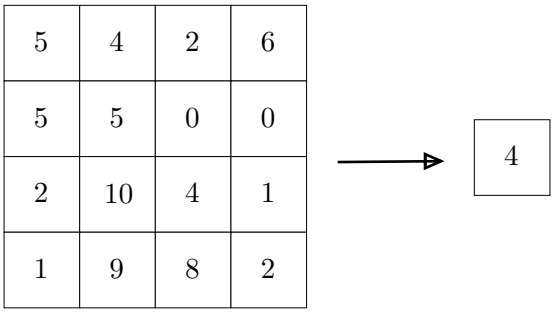


Figure 2.16: Example of a global average pooling operation. Rather than downsampling, the result is the average of each value in the 4×4 activation map.

2.3.6 Batch Normalization Layers

While initial CNNs such as AlexNet favor dropout as a regularization methode [42], dropout has since been replaced by *batch normalization* layers. Batch normalization [52] is necessary to counteract *Internal Covariate Shift* in a network. Internal Covariate Shift is an issue whenever the update of parameters in the network causes deeper layers to adapt to the new distribution of their inputs (mainly slowing the training). Solving Internal Covariate Shift can be done using a trainable normalization layer, the batch normalization layer, that normalizes the input. For a n-dimensional input $x = (x^{(1)}, x^{(2)}, ...x^{(n)})$, equation 2.23 [52, p. 3] gives the normalization of that input for a mean and variance calculated over the batch. Equation 2.24 [52, p. 3] then gives the calculation that batch normalization layer performs. Note that the trainable parameters γ and β allow the network to scale and shift the normalized input.

$$\hat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}\left[x^{(k)}\right]}{\sqrt{\mathrm{Var}\left[x^{(k)}\right]}} \tag{2.23}$$

$$y^{(k)} = \gamma^{(k)}\hat{x}^{(k)} + \beta^{(k)} \tag{2.24}$$

The advantages of batch normalization, as stated in [52], are two-fold. First, because it prevents small changes in parameters from having large influences deeper in the network, the

learning rate can be increased. A second advantage is the regularization that is provided. This is because the entire batch influences the input value $\hat{x}^{(k)}$, which is found to have a regularizing effect[4].

2.4 Convolutional Neural Network Architectures

Besides discussing the basic building blocks used in CNNs, it is important to examine the network architectures that will be used in this research. Each of these networks does not simply use standard convolution or pooling layers, but rather adds refinements to these layers for improved performance.

2.4.1 ResNet

As a popular and powerful CNN, the ResNet architecture [6] is commonly used for deep learning image recognition tasks. When ResNet was initially presented, it was designed to handle a degradation problem (vanishing gradients) introduced in the growing deep learning structures. With for example VGGNet [45] as a successor of [42], a general tendency was to simply add more layers in an attempt to increase accuracy. In [6] however, it is proven that adding layers causes a higher training error rather than increased accuracy. That is why they introduce *residual learning* in an effort to keep allowing for increased network depth.

Residual learning

At the base of the architecture of ResNet lies the method of residual learning. This method entails that a number of layers, greater than two, would need to learn the mapping $H(x)$ where x is the input of those layers. Corresponding to [6], these layers could also learn the mapping $F(x) = H(x) - x$ instead of $H(x)$. To ensure that the final output still is $H(x)$, the output of those layers will then have to be added with x. The output is then $F(x)+x$, which should result in the same as $H(x)$. Figure 2.17 demonstrates this principle.
If it is assumed that identity mappings (adding the x) are optimal, then it is reasoned by [6] that it is easier to push $F(x)$ to zero than to push $H(x)$ to x. Using the residual technique, training the network would in this case just be bringing the weights of the layers to zero. In the experiments of [6], it is shown that the learned values for $F(x)$ are small (close to zero), indicating that the assumption is true enough for the residual learning to improve accuracy.
The practical implementation of the technique of residual learning is shown in figure 2.17 and indicated in equations 2.25 and 2.26 [6, p. 3]. $F(x, W_i)$ is the residual mapping of the input x with the weights W_i. By simply adding x to the result of $F(x, W_i)$, the identity mapping is achieved. This only works if the dimensions of x and y are the same however (eq. 2.25), otherwise a linear projection W_s is necessary (eq. 2.26).

$$y = F\left(x, W_i\right) + x \tag{2.25}$$

$$y = F\left(x, W_i\right) + W_s x \tag{2.26}$$

Residual Networks

Using the practical implementation of residual learning, [6] designed a number of architectures called *Residual Networks* or ResNets. They are usually identified as ResNet-n, where n would be 18, 34, 50, 101 or 152 depending on the number of layers in the network. ResNet-18 and ResNet-34 perform identity mapping every two layers, while the other architectures do that every 3 layers instead. Noteworthy is that the convolution windows used for the first are all of size 3×3, while the latter have windows in the order of 1×1, 3×3 and 1×1 for the three layers

[4] Otherwise, $\hat{x}^{(k)}$ would not be influenced by other values in its batch.

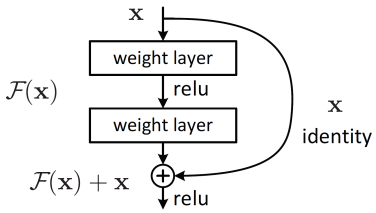


Figure 2.17: Base building block of residual learning [6, p. 2].

respectively. All architectures use batch normalization and max pooling between activation maps and global average pooling to feed the final activation maps into a softmax function.

2.4.2 MobileNetV2

Most deep learning architectures seek to grow ever larger resulting in heavy computational and memory requirements. This causes them to be unusable by mobile phones or embedded platforms. MobileNet [53] and MobileNetV2 [5] strive to resolve this issue by providing a smaller, more efficient solution. This is done using several innovations (in this section the main focus lies on MobileNetV2).
Firstly, they use *Depthwise Seperable Convolutions* rather than standard convolutions. Depthwise Seperable Convolutions occur in two steps. In the first step, the depthwise convolution, a single filter is applied to each input channel. In the next step, the pointwise convolution, a 1×1 filter combines the outputs of the first step. Standard convolution performs both tasks in one step, causing a significant increase in computational cost and model size [53].
Secondly, linear bottlenecks reduce the number of input channels for each convolution layer. A bottleneck layer is a set of layers such as for ResNet-50, ResNet-101 or ResNet-152: 1×1 followed by 3×3 followed by another 1×1 layer. The 1×1 layers are used to reduce the number of channels the 3×3 layer needs to process. If the number of input channels would be A, the first 1×1 layer would produce $B < A$ activation maps that the 3×3 receives as input. The last 1×1 layer would then return the number of input channels to A. This procedure is more efficient than simply using a series of convolution layers. In [5], more in-depth explanation and arguments are given to substantiate this claim. They also refer to the ratio between A and B as the *expansion ratio*.
Finally, in [5] the intuition that the bottleneck part (the 3×3 part) contains all useful information while the 1×1 layers (*expansion layers*) simply provide resizing of the dimension is followed. Rather than providing shortcuts between expansion layers, as done for residual learning, shortcuts between the bottlenecks are used instead. This method of working is referred to as *inverted residuals*.
Using all these techniques, MobileNetV2 is a memory-efficient network that retains accuracy and is computationally more efficient than other state-of-the-art networks. To demonstrate this, [5] compare their performance against NasNet-A [54] (a state-of-the-art architecture) for training on the ImageNet [44] dataset. Where NasNet needs 5.3 million parameters and has 564 million Multiply-Adds, MobileNetV2 (only) requires 3.4 million parameters and 300 million Multiply-Adds while the obtained accuracy for the networks is similar (MobileNetV2 even scores a little higher).

2.5 Related work

In research, it is important to be able to compare results with relevant work whenever possible. Therefore, in this section a number of other projects working on wildlife classification based on camera trap data are discussed.

2.5.1 Chen et al.

In [3], Guobin Chen et al. describe one of the first attempts to automate animal classification for camera trap data on a large scale. They first use Ensemble Video Object Cut (EVOC) [55] to extract the Region Of Interest (ROI) from the images. The resulting ROIs are then used for image classification using two machine learning algorithms: Bag of visual Words (BOW) [56] and Deep CNN. Figure 2.18 visualises the used Deep CNN. Their dataset consists out of 14346 training examples and 9530 testing examples for 20 North American animal species. Using this dataset to train and test their algorithms, they reported an accuracy of 33.507% for the BOW and an accuracy of 38.315% for the Deep CNN. While these accuracies are quite low, these results at least indicate that a Deep CNN is more suited for the classification task, compared to a BOW. The authors expect that collecting more camera trap data will improve the species recognition performance.

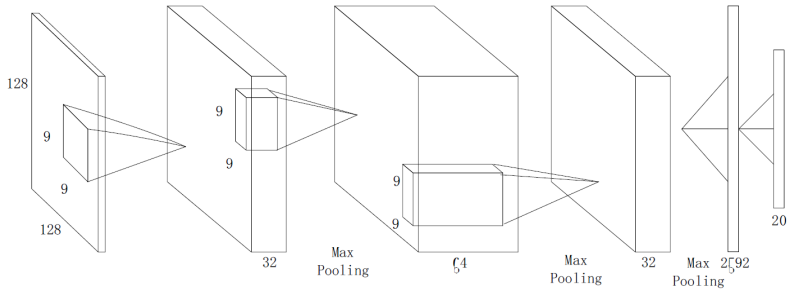


Figure 2.18: The Deep CNN used by Chen et al. to classify cropped camera trap data [3, p. 860].

2.5.2 Gomez et al.

More recent and more relevant to this research is the work of Gomez et al. [57]. Its relevancy in the first place comes from working on the Snapshot Serengeti dataset, which is the dataset used in this project. In addition, they examine both different versions of this dataset and different CNN architectures. More specifically, they use an unbalanced version and a balanced version of the dataset to demonstrate the effect of unbalance in the dataset. Unbalance in this case means that some species are far more present in the dataset than other species. Additionally, a version of the data with only images where an animal is present in the foreground is used. Finally, a fourth version only uses manually segmented images providing a good view of the visible (part of the) animal to simulate results in case of an ideal segmentation algorithm. These datasets are labelled *D1* through *D4*. The used CNNs are AlexNet, VGGNet, GoogLeNet, ResNet-50, ResNet-101, ResNet-152 and finetuned versions of AlexNet and GoogLeNet, respectively labelled *A* through *H*. These networks are trained using transfer learning. Transfer learning is repurposing a network that is pre-trained on a certain dataset for a different dataset. In [57], this is done in two ways. The finetuned networks retrain all layers on the Serengeti dataset. On the contrary, the other six networks do not retrain the high-level convolution layers, but use these as an input for a linear classifier.
Figure 2.19 gives the results for the conducted experiments. While for *D4* deceptively high Top-1 (88.9%) and Top-5 (98.1%) accuracies are reported[5], it is important to keep in mind that these results are attained using manually segmented images. As the algorithm to automatically segment these images does not currently exist, such accuracies cannot be expected in a regular situation. Accuracies for less tailored versions of the dataset are lower, but still significantly better than those reported in [3]. When using network *E* in the situation of [3], higher accuracies have indeed been reported. More importantly, these experiments indicate that the best results are always obtained by the ResNet architectures for this dataset.

[5]The significance of Top-1 and Top-5 accuracies is discussed in section 3.3.6.

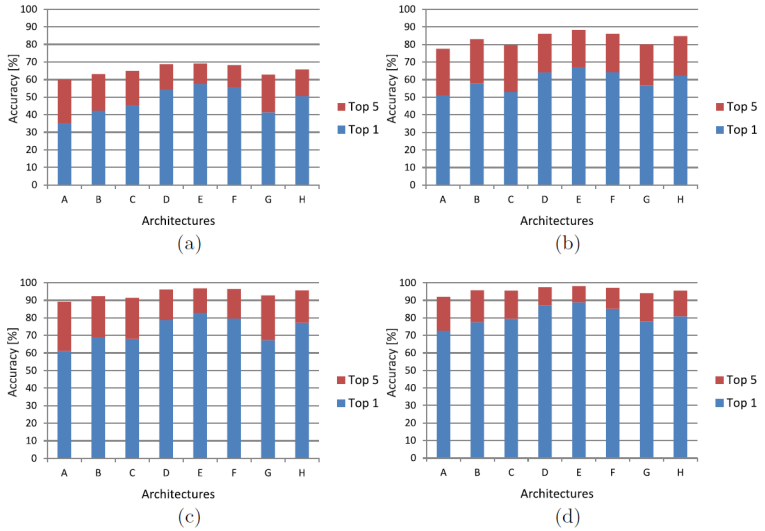


Figure 2.19: Results of the experiments performed in [57, p. 10]. Graphs *a* through *d* represent the results for the datasets *D1* through *D4*, while each bar in the graphs presents the result of a specific network on that dataset.

2.5.3 Norouzzadeh et al.

In [58] additional research has been conducted on the Snapshot Serengeti dataset. Not only is the classification of the images researched, but also other classification tasks have been implemented: counting animals, describing the animal activity and detecting the presence of young animals. Moreover, they claim to have achieved better results than [57], as they report a higher accuracy and have not limited the dataset in the ways [57] did. Their classification uses a two step approach. In the first step, the images containing animals are detected. The reported accuracy for this task is $\geq 95.8\%$ for all used networks. Hereafter, images containing animals are classified. Table 2.1 gives the results of this classification for the various CNNs used. Note that the classification of animals is tested separately of the classification into blanks and animals. Consequently, the reported accuracies are calculated solely with the performance on images containing animals (which means it is assumed that all images without animals are filtered during the previous step). The next steps, respectively counting the animals and detecting attributes (whether the animals are eating,...) will not further be discussed here. As shown in table 2.1, the accuracies are significantly higher than in [57]. [58] even claims that they have reached the same approximate accuracy as the human volunteers responsible for labelling the dataset. At any rate, these excellent results provide a good point of reference to work with. Disregarding the ensemble, the ResNet architectures again reach the highest accuracy for the classification task.

2.6 Solar irradiation

In order to provide a CNN with additional insight, it might be beneficial to use metadata concerning the processed images to provide additional training features. This section discusses a possible metadata feature, namely the amount of sunlight in an image. To do this, the clear-sky model of the *American Society of Heating, Refrigerating and Air-Conditioning Engineers* (ASHRAE) [59] can be used to approximate the solar irradiance incident on a specific location at a specific time. This section will give the empiric equations provided by [59] that allow to calculate the solar irradiation for a given location at a given time.

Table 2.1: Result of Norrouzadeh et al. [58]. NiN stands for Network in Network [51], the ensemble is created using all the other networks.

Architecture	Top-1(%)	Top-5(%)
NiN	92.0	98.4
AlexNet	92.4	98.5
VGG	92.4	98.6
GoogLeNet	93.0	98.6
ResNet-18	93.2	98.6
ResNet-34	93.3	98.6
ResNet-50	93.6	98.4
ResNet-101	93.8	98.8
ResNet-152	93.8	98.8
Ensemble	94.9	99.1

2.6.1 Solar irradiation

For all solar irradiation calculations, the solar constant $E_{sc} = 1367\ \mathrm{W/m^2}$ as defined in [60, p. 10] will be used. Equation 2.27 then gives a way to calculate the *extraterrestrial solar radiance* E_0 [59, p. 14.8]. n in this case is the day of the year (January 1 would be $n = 1$, December 31 would be $n = 365$ for a non-leap year).

$$E_0 = E_{sc}\left(1 + 0.033\cos\left(360^\circ\frac{n-3}{365}\right)\right) \tag{2.27}$$

2.6.2 Time

In order to be able to perform the calculations that are defined further on in this section, it is necessary to find a value for the *Apparent Solar Time* AST. If the Apparent Sun can be defined as the Sun as it appears to the observer [61], the AST is the time corresponding to that apparent location. The AST can be found using equation 2.28 [59, p. 14.8].

$$AST = LST + \frac{ET}{60} + \frac{LON - LSM}{15} \tag{2.28}$$

This equation requires the *Local Standard Time* LST in decimal hours, the *equation of time* ET in minutes, the longitude LON to the East of Greenwich in degrees and the *longitude of the local standard meridian* LSM to the East of Greenwich in degrees. Equations 2.29 and 2.30 give a way to calculate ET [59, p. 14.8], equation 2.31 [59, p. 14.8] gives a way to calculate LSM and equation 2.32 [59, p. 14.9] finally gives a way to calculate LST for a given *Daylight Saving Time* DST. In equation 2.31, TZ stands for the time zone.

$$\begin{aligned} ET = 2.2918[&0.0075 + 0.1868\cos(\Gamma) - 3.2077\sin(\Gamma) \\ &- 1.4615\cos(2\Gamma) - 4.089\sin(2\Gamma)] \end{aligned} \tag{2.29}$$

$$\Gamma = 360^\circ\frac{n-1}{365} \tag{2.30}$$

$$LSM = 15TZ \tag{2.31}$$

$$LST = DST - 1 \tag{2.32}$$

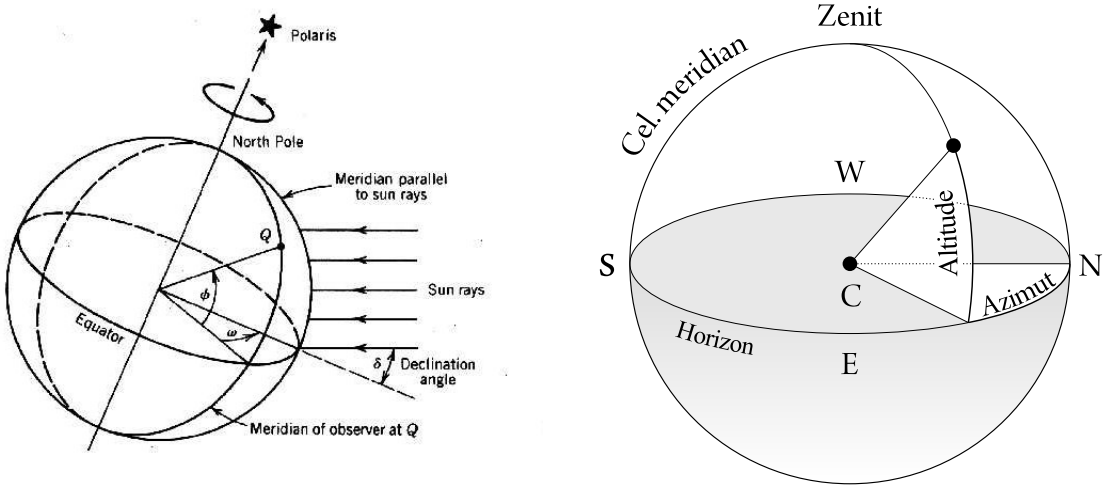
2.6.3 Astronomy parameters

It is necessary to know the current position of the Sun, relative to Earth, to be able to model the solar energy reaching Earth. The first necessary value is the *solar declination* δ, the angle between the Sun-Earth line and Earth's celestial equator [62]. An approximate value for δ in degrees can be found using equation 2.33 [59, p. 14.9].

$$\delta = 23.45 \sin\left(360^\circ \frac{n + 284}{365}\right) \tag{2.33}$$

If the declination represents the angular displacement perpendicular to the celestial equator (the North-South angle), the *hour angle* ω represents the angular displacement relative to the local meridian (the East-West angle) as seen in figure 2.20a. Equation 2.34 [59, p. 14.9] gives the calculation for ω.

$$\omega = 15(AST - 12) \tag{2.34}$$



(a) (b)

Figure 2.20: Left: The meaning of the solar declination δ and the hour angle ω [63, p. 2]. Note that the ϕ depicted here is not the azimuth angle. Right: Graphical overview of the meaning of solar altitude and azimuth [64]. In this case, the point connected to the observer in C is the Sun.

Finally, the *solar altitude* β and the *azimuth angle* ϕ can be found using respectively equations 2.35 and 2.36 [59, p. 14.9]. Figure 2.20b gives an overview of what these parameters are.

$$\beta = \arcsin\left(\cos(L)\cos(\delta)\cos(\omega) + \sin(L)\sin(\delta)\right) \tag{2.35}$$

$$\phi = \arcsin\left(\frac{\sin(\omega)\sin(\delta)}{\cos(\beta)}\right) \tag{2.36}$$

2.6.4 Air mass and clear-sky irradiation

One of the final parameters necessary for the calculation of the solar irradiation is the *relative air-mass* m. One way to calculate m is equation 2.37 introduced in [59, 65]. In this equation, β is the solar altitude.

$$m = \frac{1}{\sin(\beta) + 0.50572(6.07995 + \beta)^{-1.6364}} \tag{2.37}$$

Using this value, equations 2.38 and 2.39 respectively give the beam normal irradiance E_b and the *diffuse horizontal irradiance* E_d [59, p. 14 20], two values that together give an approximation of the solar irradiation incident on a certain location.

$$E_b = E_0 \exp -\tau_b m^{ab} \tag{2.38}$$

$$E_d = E_0 \exp -\tau_d m^{ad} \tag{2.39}$$

In equations 2.38 and 2.39, the values for the *beam air mass coefficient ab* and the *diffuse air mass coefficient ad* can be calculated using equations 2.40 and 2.41 [59, p. 14 20]. τ_b and τ_d finally are location-dependent parameters.

$$ab = 1.454 - 0.406\tau_b - 0.268\tau_d + 0.021\tau_b\tau_d \tag{2.40}$$

$$ad = 0.507 + 0.205\tau_b - 0.080\tau_d - 0.190\tau_b\tau_d \tag{2.41}$$

The solar irradiation calculated in this section is the approximate clear-sky irradiation, implying that this approximation is only legitimate under a clear sky. Whenever clouds or other dimming phenomena are present, the accuracy of the model will falter.

2.7 Lunar irradiation

As the model used in the previous section only provides information about the Sun's influence on a location, its output data only provides useful values during the day. All solar irradiation values during the night are zero, as there is no direct sunlight at night. That is not a correct representation of reality however, as that would mean no natural light is present at night. On the contrary, natural light is present in the form of sunlight reflected on the surface of the Moon. Taking this source of natural irradiation into account could prove beneficial for a CNNs learning capability. This section discusses the basic properties of lunar irradiation and possible ways to calculate the Moon's influence on Earth.

2.7.1 Lunar phase cycle

Key to understanding the influence of the Moon is sufficient insight in its different phases. Figure 2.21 from [66] gives a simple and clear representation of how these phases function. Assume that the Moon starts in position 1. In this point, *new Moon*, the surface of the Moon facing Earth is completely dark, as no light from the Sun directly reaches it. The Moon's orbital longitude *l"* equals the Sun's longitude λ. Starting in this point, the Moon's position gradually changes until the Moon is in quadrature. This is the *first Quarter* at point 3. The Moon's surface facing Earth is now partially lit by incident sunlight. Next, position 4 indicates a *full Moon* at which point the Moon appears to be completely lit. In position 5 the Moon is again in quadrature, having reached its *third* or *final quarter*. More concretely, the phases new Moon, first quarter, full Moon and final quarter are when *l"* and λ differ by respectively: 0°, 90°, 180° and 270° [67]. There are two ways to express the phase of the Moon. One is the age of the Moon D, expressed in degrees (0° to 360°). Another is the phase F, expressed in %[6]. Equation 2.42 provided in [66] demonstrates the relation between these two values.

$$F = \frac{1}{2}(1 - cos(D)) \tag{2.42}$$

As lunar irradiation on Earth is dependent on the sunlight reflected by the Moon, and as that reflection is dependent on the current phase of the Moon, it is clear that it is essential to be able to accurately obtain that phase to calculate the lunar irradiation on Earth.

[6]More specifically, F can be viewed as the fraction of the lunar that disc that is illuminated [66]. This can be expressed as a value between zero (no illumnation) and one (fully illuminated), but is sometimes referred to as a percentage. Unless otherwise specified, F will be viewed as a value between zero and on in this project.

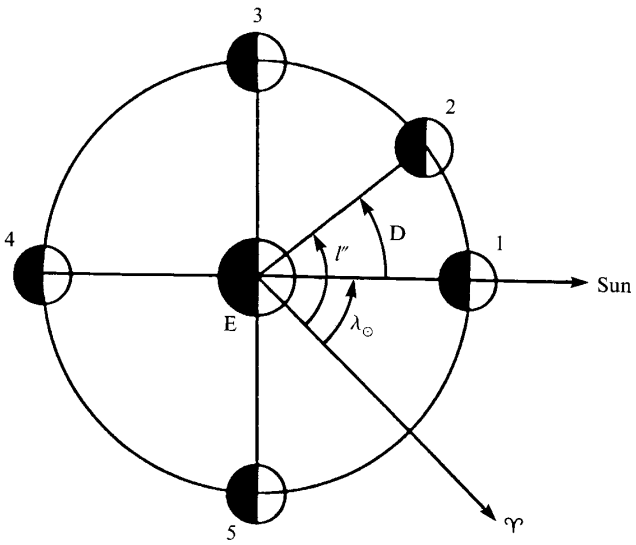


Figure 2.21: Different phases of the moon as presented in [66]. The Moon orbits Earth (E), its phases going from New Moon (1) to Full Moon (4) to return to New Moon after one cycle. *l"* is the Moon's true orbital longitude, D is the Moon's age, λ the Sun's longitude and Υ is the direction of the first point of Aries, a reference.

2.7.2 Calculation of the lunar phase

While accurately calculating the lunar phase is no easy task (due to the complex Earth-Moon-Sun relations), several relatively simple approaches exist. One of the first major contributions to astronomical calculations is given in [68], where simple algorithms are given to calculate many important astronomical values. One of these algorithms provides a way to calculate the date on which a new/full Moon or first/final quarter will occur. Since this algorithm does not allow an easy calculation of the current phase for a given date, and as it is quite outdated, it is necessary to use a more recent algorithm. The algorithm presented in [66] provides sufficient accuracy while maintaining a relative ease of computation. This section will briefly introduce the calculations given in [66]. In order to calculate the phase of the Moon, it is necessary to know the locations of both the Sun and the Moon, relative to Earth. This principle is demonstrated in figure 2.22. That means that calculations for both the Sun and the Moon have to be performed.

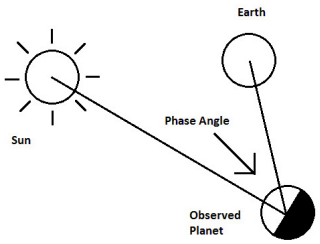


Figure 2.22: Simple demonstration of what the phase angle of a celestial body (in this case the Moon) is [69].

Julian Day and epoch

Many astronomical calculations use *Julian Day* (JD) as a unit of time, rather than local time or *Universal Time* (UT). Julian Day basically is the number of days since the beginning of the year 4712 BC [68], and allows for an easy interpretation of differences in days between certain points in time. Algorithm 1 is the conversion algorithm as provided in [66] that converts a date in UT to a value in JD. It is important to note that the day in this case is a *decimal day*, meaning the hours, minutes and seconds are processed into the notation of the day. Equation 2.43 gives one

Algorithm 1 Universal Time to Julian Day

procedure UT2JD
  $y = year, m = month, d = decimal\ day$
  if $m = 1$ or $m = 2$ **then**
    $y' = y - 1$
    $m' = m + 12$
  else
    $y' = y$
    $m' = m$
  if $date < 1582\ October\ 15$ **then**
    $A = \text{TRUNC}(y'/100)$
    $B = 2 - A + \text{TRUNC}(A/4)$
  else
    $B = 0$
  if $y' < 0$ **then**
    $C = \text{TRUNC}((365.25 \times y') - 0.75)$
  else
    $C = \text{TRUNC}(365.25 \times y')$
  $D = \text{TRUNC}(30.6001 \times (m' + 1))$
  $JD = B + C + D + d + 1720994.5$

simple way to get this decimal day. If *day* is the integer value for the day of the month, *hour*, *minute* and *second* are the time values of that day that need to be added to get the decimal value.

$$d = day + \frac{hour}{24} + \frac{minute}{24 * 60} + \frac{second}{24 * 3600} \tag{2.43}$$

In addition to the previous point, the use of the *TRUNC* function needs to be explained. Essentially, this function takes a floating point value and only retains the integer part of that value. TRUNC(3.99) would for example be 3 while TRUNC(-3.99) would be -3.

Another parameter that is often used in astronomy is T, the number of Julian centuries since the beginning of a reference epoch. Considering a Julian year is exactly 365.25 days [70], equation 2.44 gives the usual way to calculate T [66].

$$T = \frac{JD - epoch}{36525} \tag{2.44}$$

In this equation *JD* is the point in time for which T must be found in Julian Day, and *epoch* is the reference epoch in Julian Day.

Position of the Sun

To be able to calculate the position of the Sun, a number of constants need to be determined. Important to note is that all values are calculated in reference to epoch 2010 January 0.0 (JD = 2,455,196.5). In addition, all angular values are in degrees. Finally, all equations in the following sections are given in [66]. Taking these things into account, ϵ_g is the Sun's mean ecliptic longitude at the epoch. This value gives the starting point of the Sun if its orbit would be circular rather than elliptic. Equation 2.45 gives a way to calculate it.

$$\epsilon_g = 279.6966778 + 36000.76892T + 0.0003025T^2 \tag{2.45}$$

Here T is the number of Julian centuries since the reference epoch. In a similar fashion, $\overline{\omega}_g$ and e are respectively the longitude of the Sun at perigee (nearest to Earth) and the eccentricity

of the Sun-Earth orbit. Equations 2.46 and 2.47 give the formulas to calculate these parameters.

$$\overline{\omega}_g = 281.2208444 + 1.719175T + 0.000452778T^2 \tag{2.46}$$

$$e = 0.01675104 - 0.0000418T - 0.000000126T^2 \tag{2.47}$$

Taking these constants, the mean anomaly M can be calculated. This is the angle the Sun would have travelled since perigee if it moved in a circle at constant speed. If t is the number of days since the epoch, equation 2.48 gives the calculation of M.

$$M = \frac{360}{365.242191} \cdot t + \epsilon_g - \overline{\omega}_g \tag{2.48}$$

By adding some correction terms to M, the longitude of the Sun λ can be found. This value, given in equation 2.49, is the final solar value necessary for lunar phase calculation[7].

$$\lambda = M + \frac{360}{\pi} e \sin(M) + \overline{\omega}_g \tag{2.49}$$

Position of the Moon

In addition to the position of the Sun, the position of the Moon is vital for correct phase calculations. If the same epoch is used here, table 2.2 provides the constant values that are necessary in all calculations in this section. Both M as well as λ, as found in calculations 2.48 and 2.49, are required in some of the following equations. Taking the constants provided in table 2.2, equations 2.50, 2.51 and 2.52 respectively give the mean longitude l, the mean anomaly M_m and the ascending node's mean longitude N for the Moon.

$$l = 13.1763966D + l_0 \tag{2.50}$$

$$M_m = l - 0.1114041D - P_0 \tag{2.51}$$

$$N = N_0 - 0.0529539D \tag{2.52}$$

Table 2.2: Constants as defined for epoch 2010 January 0.0 in [66]. The units for all constants are degrees.

l_0	91.929336
P_0	130.143076
N_0	291.682547
i	5.145396

Calculating the corrections for evection E_v and the annual equation A_e in equations 2.53 and 2.54, the Moon's anomaly can be corrected (equation 2.55).

$$E_v = 1.2739 \sin(2(l - \lambda) - Mm) \tag{2.53}$$

$$A_e = 0.1858 \sin(M) \tag{2.54}$$

$$M'_m = M_m + E_v - A_e - 0.37 \sin(M) \tag{2.55}$$

After again applying a number of corrections, the Moon's corrected longitude l' can be found (equation 2.56).

$$l' = l + E_v + 6.2886 \sin(M'_m) - A_e + 0.214 \sin(2M'_m) \tag{2.56}$$

Finally, for calculating the lunar phase, the Moon's true orbital longitude l" must be found. This value can be found using equation 2.57.

$$l'' = l' + 0.6583 \sin(2(l' - \lambda)) \tag{2.57}$$

[7] Additional calculations can be made to find more parameters concerning the Sun's position [66]. As these are unnecessary for this research, they are not discussed here.

Lunar phase

When enough information of the Sun's and Moon's positions has been collected, the lunar phase (angle) can be calculated. Taking the Moon's longitude l'' and the Sun's longitude λ, the age of the Moon D can be calculated. This is, as previously mentioned, simply the difference between the two angles. That is also why D is actually represented in degrees rather than a unit of time. Equation 2.58 gives this difference, and equation 2.42 then allows for the calculation of the actual phase F.

$$D = l'' - \lambda \tag{2.58}$$

2.7.3 A model for lunar irradiation

Having calculated the phase of the Moon for a current point in time, the lunar irradiation incident on Earth must be found. While much research is conducted concerning the surface of the Moon, not many mathematical models for lunar irradiation exist. In [71] one of the first attempts to calculate moonlight illuminance on Earth is done. While they achieve fairly accurate results, the project is not easily reproducible as their descriptions are not very in-depth and their used sources are outdated. One example of this is that while explaining what the developed program does, no sufficient explanation of how this is done is given. Similarly, [72] proposes a model to calculate moonlight brightness, published in 1991. But as they report an accuracy between 8% and 23%, this model will not be further discussed here. Finally, a more recent approach to calculate lunar irradiation is given in [73]. More specifically, the *Top-Of-Atmosphere* TOA irradiance, E_{TOA}, is calculated based on a number of statements. Particularly, they state that lunar spectral irradiance is subject to:

1. Changes in Sun-Earth, Moon-Earth and Sun-Moon distances.
2. Changes in lunar phase.
3. Changes in sunlight incident on the moon.
4. Influence of the surface of the Moon (craters, ...).

It is worth noting that E_{TOA} represents the lunar irradiation in the top of Earth's atmosphere, not the lunar irradiation incident on Earth's surface. Therefore, it is assumed that a fraction of this irradiation reaches the surface (in [73] one way to calculate this is very briefly described). As is argued in section 3.2, the exact value must not be known, which is why it will not be calculated. Taking all of this information into account, a model to calculate E_{TOA} can be defined.

Lunar magnitude and phase function

Based on [74], [73] define a lunar magnitude m, given in equation 2.59 [73, p. 2319].

$$m(\theta_p, \lambda_s) = a(\theta_p) - b(\theta_p)\lambda_s \tag{2.59}$$

Here θ_p is the lunar phase angle, and λ_s is the wavelength of the light that is inspected in μm. Note that the subscript s has been added to avoid confusion with the solar longitude λ, this s signifies the spectral character of the irradiation calculated in this model. Another important remark is that while the lunar age D presented in the previous section is an angle between 0°and 360°, θ_p is an angle that varies between 0°and 180°. Finally, the values for $a(\theta_p)$ and $b(\theta_p)$ have to be interpolated or extrapolated from table 2.3 given in [73]. Using the value for m calculated in equation 2.59, a value for the phase function $f(\theta_p, \lambda_s)$ can be found using equation 2.60 [73, p. 2319]. The resulting value for f can then be used in the following calculations for lunar irradiation.

$$f(\theta_p, \lambda_s) = 10^{-0.4m} \tag{2.60}$$

Table 2.3: Values of the functions $a(\theta_p)$ and $b(\theta_p)$ for the given input lunar phase angles in degrees [73, p. 2320]. They are called *linear fit coefficients*. Values for other phase angles need to be found through interpolation and extrapolation.

θ_p	$a(\theta_p)$	$b(\theta_p)$
10°	0.30805	0.0652190
20°	0.61820	0.1336200
30°	0.92169	0.1989200
40°	1.23130	0.2668300
50°	1.50230	0.2961900
60°	1.76490	0.3078300
70°	2.04800	0.3204200
80°	2.36050	0.3320600
90°	2.72480	0.3540200
100°	3.15660	0.3995500
110°	3.66680	0.4609600
120°	4.27780	0.5570100

Lunar albedo

In addition to the phase function, a value for the *lunar spectral albedo* α, the measure of whiteness or reflection, is necessary. Equation 2.61 gives α for a wavelength λ_s in μm in the spectrum. The values for A_i can be found in table 2.4 [73, p. 2321].

$$\alpha(\lambda_s) = \sum_{i=1}^{n} A_i \lambda_s^i \tag{2.61}$$

Taking α for a certain wavelength, the reflected portion of the solar irradiation incident on the Moon can be found. If E_m would be the sunlight incident on the surface of the Moon, the reflected light L_m would be given by equation 2.62 [73, p. 2320].

$$L_m = \frac{\alpha E_m}{\pi} \tag{2.62}$$

Mean lunar TOA irradiation

When α and $f(\theta_m, \lambda_s)$ are found, taking a number of constants given in table 2.5, equation 2.63 gives a way to calculate the mean Top-Of-Atmosphere lunar irradiation $\overline{E}_{TOA}$ [73, p. 2323].

$$\overline{E}_{TOA} = \alpha E_0 \left(\frac{\overline{R}_{se}}{\overline{R}_{sm}}\right)^2 \left(\frac{r_m}{\overline{R}_{me} - r_e}\right)^2 f(\theta_p) \tag{2.63}$$

This final equation allows to approximate the lunar irradiation on Earth. It helps to obtain a better understanding of the influence of the moonlight in an image.

Table 2.4: Coefficients used in equation 2.61 as found in [73, p. 2321].

Coefficient	$0.30 < \lambda_s < 0.60 \mu m$	$0.60 \geq \lambda_s < 1.20 \mu m$
A_0	-0.04944	-0.7317
A_1	0.4406	3.621
A_2	-0.3150	-5.656
A_3	0.1084	3.934
A_4	n.a.	-0.9999

Table 2.5: Astronomy constants as found in [73, p. 2318]. Note that E_0 defined here is different from the E_0 given in section 2.6.1. While E_0 practically represents the same value as E_{sc} in in section 2.6.1, not every model uses exactly the same value for this solar constant. To be exact, the value for the solar constant as defined in [60] is 1367 ± 7 W/m^2. In [73], the authors decided to use 1361 W/m^2. The reason for choosing this specific value is not specified.

Name	Symbol	Value
Sun Radius	r_s	695508 km
Moon Radius	r_m	1737.4 km
Earth Radius	r_e	6378.14 km
Moon-Earth Perigee Distance	$R_{me,p}$	356371 km
Moon-Earth Apogee Distance	$R_{me,a}$	406720 km
Sun-Earth Perihelion Distance	$R_{se,p}$	147088067.2 km
Sun-Earth Aphelion Distance	$R_{se,a}$	152104233.4 km
Astronomical Unit	AU	14959 870.7km
Mean Sun-Earth Radius	$\overline{R}_{se}$	149598022.6 km
Mean Moon-Earth Radius	$\overline{R}_{me}$	384.401 km
Total Solar Irradiance	E_0	1361 W/m^2 (at 1 AU)

2.8 Summary

As seen in this chapter, taking the knowledge of machine learning and using CNNs allows for the classification of the Snapshot Serengeti dataset. More specifically, the MobileNetV2 and ResNet-50 architectures that have been discussed will be used for experimentation as described in the next chapter. Additional knowledge can be obtained from research performed by other authors such as Norouzzadeh et al.. Combining this classification with the calculation of solar and lunar irradiation might then provide more insight in the functioning of CNNs and their ability to recognize metadata features. More precisely, the brightness in an image can be correlated with the sunlight and the moonlight incident at the image's location at the time of capture. The next chapter will use all this information to conduct experiments.

Chapter 3

Method

After explaining the underlying concepts of machine learning and metadata calculation, the research steps taken are discussed. First, this chapter explains how the dataset has been used and how the corresponding metadata can be found and used. Next, it gives the steps that are taken to train classification and regression networks. All this information is required to fully understand the results and discussion in the next chapter.

3.1 Description of the Snapshot Serengeti dataset

The Snapshot Serengeti dataset comprises 1.2 million image sets of 1-3 images also referred to as capture events. It contains labeled data for images of 48 different animal species in the Serengeti National Park including the number or count, behaviour, and presence of young. The Snapshot Serengeti team hosts the entire dataset on Amazon Web Services using Amazon's Simple Storage Service (S3). The labeled data is available in the CSV files from [7]. Based on the information in the images, raw classification data and consensus data CSV files, all animal species images and a large subset of the misfires or blank images were downloaded. While the species images are used to train CNNs to recognize animal species, the blank images together with the species images allow training CNNs that detect the presence of animals.

Table 3.1 shows that the dataset comprises 805,510 animal species images and 948,168 blank images. The dataset clearly suffers from class imbalance between the animal species, a typical problem in machine learning, wherein certain classes are widely represented whilst others are barely present. Above all, the wildebeest species alone corresponds to 30% of the dataset. Together with the zebra and Thomson's gazelle, these 3 species make up 62% of the dataset. The other 45 species' image counts in the dataset are much smaller with 21 species having less than 1,000 images. For example, the zorilla species only has 29 images which is practically negligible. This imbalance will hinder the neural network's capacity to equally classify each species with the same accuracy. Figure 3.1 visualizes some high quality images in the dataset wherein the animal either is clearly visible (in the case of a species image) or not present at all (in the case of blank images). It should be noted that the majority of images in the dataset are not optimal as is shown in figure 3.2, which complicates the neural network's learning process.

The dataset is split into a training, validation and test set with ratios 70%, 15% and 15% which translates to respectively 1,207,806 training images, 258,822 validation images and 258,892 test images. Furthermore, 563,801 images of the training set, 120,820 images of the validation set and 120,889 images of the test set contain animal species. Whilst making these sets, it has been made sure that the number of images from each animal species adheres to the same split ratios. Important to know is that not all labels of the species capture events in the dataset are classified with the same certainty. With the information in the Snapshot Serengeti CSV files, it is possible to remove or blacklist such image sets. More specifically, for each image set the ratio of the number of times it has been classified as blank to the total number of times it has been classified is calculated. If this species image set was classified as blank too often (compared

against a threshold), it is blacklisted[1]. The reasoning for this is that if human volunteers have trouble correctly classifying an image, it will be at least as difficult for a CNN. Rather than using these vague images to train the network, it is better to learn to classify animals using clear(er) examples. After applying the blacklist with a threshold value of 25%, 1,164,847 images are in the train set, 249,832 in the validation set, and 249,889 in the test set. For these sets, the number of animal species images is 520,935, 111,854, and 111,902 respectively.

[1]For example, if 25% of the human volunteers classifying an image indicate that they cannot see anything on that image, that image is blacklisted (if it actually contains animals).

Table 3.1: Number of images in the Snapshot Serengeti dataset.

Category	Total	Total in %
Wildebeest	243385	30.22
Zebra	147654	18.33
Thomson's gazelle	112190	13.93
Hartebeest	33852	04.20
Buffalo	33140	04.11
Human	26239	03.26
Elephant	25090	03.11
Giraffe	21735	02.70
Impala	21590	02.68
Guinea fowl	21587	02.68
Warthog	20780	02.58
Grant's gazelle	20249	02.51
Other bird	12120	01.50
Spotted hyena	10119	01.26
Female lion	8389	01.04
Eland	6681	00.83
Topi	5805	00.72
Baboon	4415	00.55
Reedbuck	4109	00.51
Cheetah	3330	00.41
Dik-dik	3325	00.41
Hippopotamus	3228	00.40
Male lion	2149	00.27
Ostrich	1869	00.23
Kori bustard	1865	00.23
Secretary bird	1209	00.15
Jackal	1154	00.14
Serval	933	00.12
Hare	890	00.11
Vervet monkey	871	00.11
Waterbuck	811	00.10
Bat-eared fox	717	00.09
Mongoose	643	00.08
Aardvark	541	00.07
Porcupine	441	00.05
Reptiles	388	00.05
Leopard	382	00.05
Bushbuck	352	00.04
Aardwolf	292	00.04
Striped hyena	271	00.03
Caracal	171	00.02
Rodents	138	00.02
Wildcat	105	00.01
Honey badger	83	00.01
Civet	67	00.01
Rhinoceros	66	00.01
Genet	61	00.01
Zorilla	29	00.00
Total	805510	100.00

Category	Total	Total in %
Species	805510	45.93
Blank or misfire	948168	54.07
Total	1753678	100.00

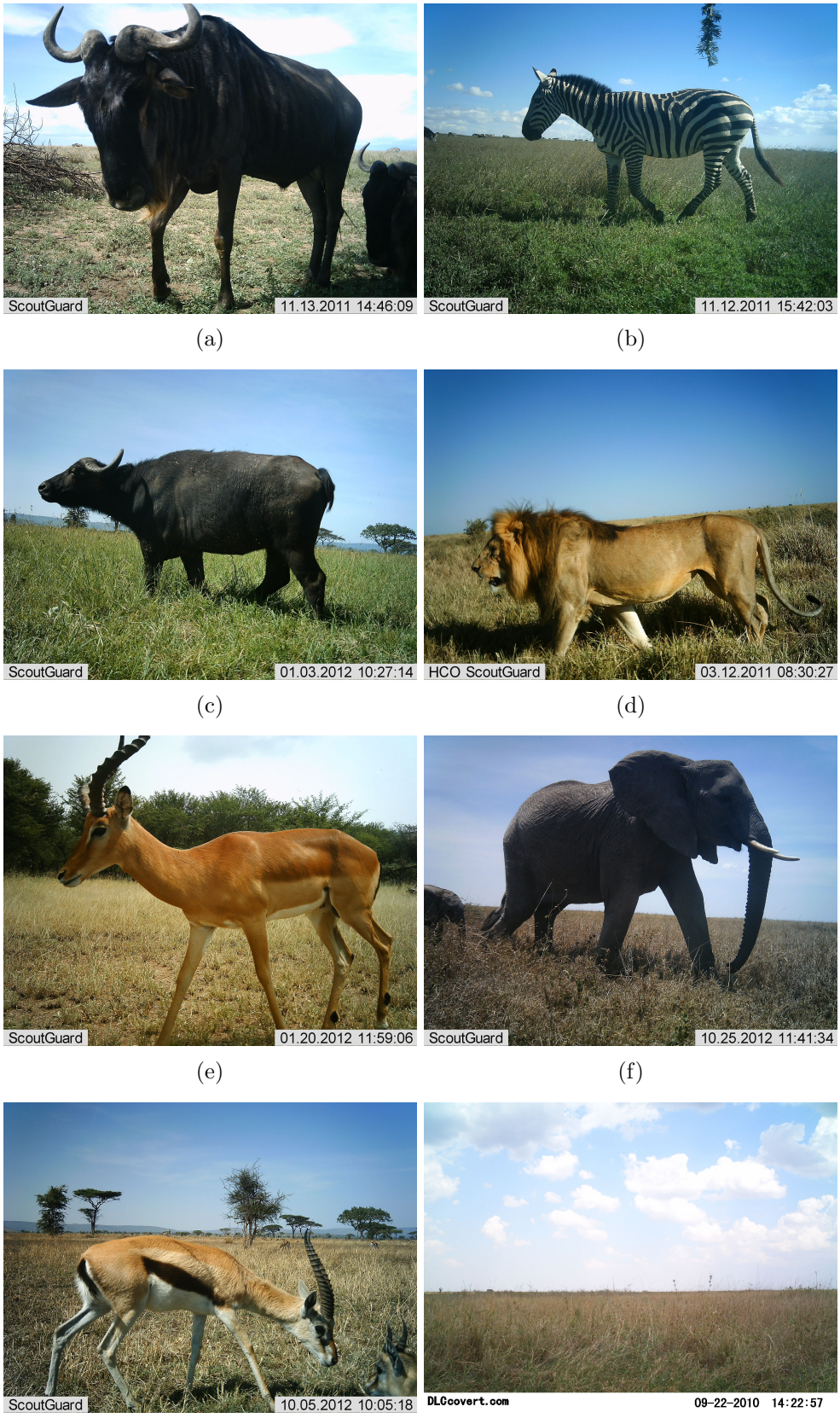


Figure 3.1: Optimal images in the Snapshot Serengeti dataset. (a) Wildebeest. (b) Zebra. (c) Buffalo. (d) Male lion. (e) Impala. (f) Elephant. (g) Thomson's gazelle. (h) Blank.

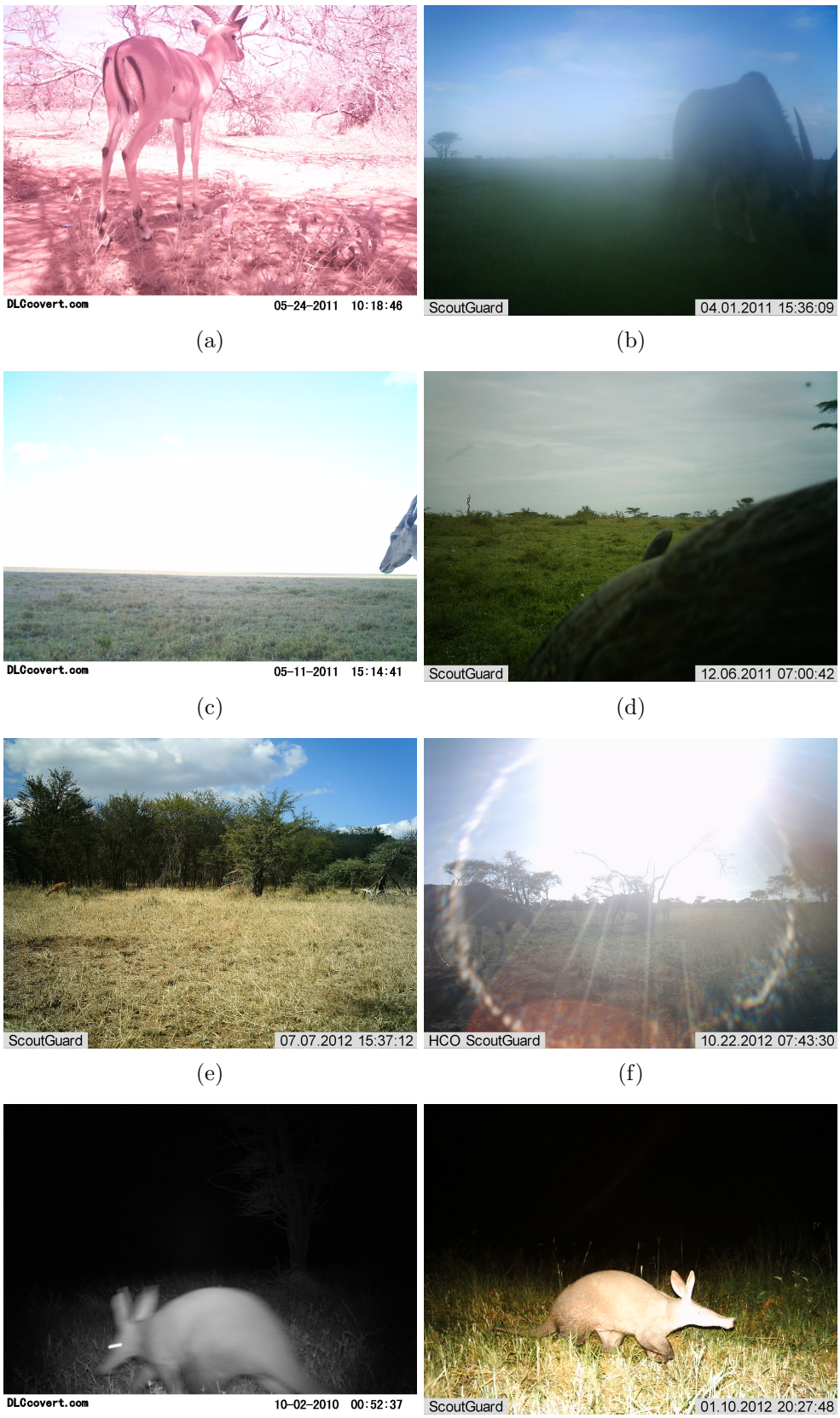


Figure 3.2: Difficult images in the Snapshot Serengeti dataset. (a) Camera error. (b) Blurred image. (c) Partially visible animal. (d) Close-up shot. (e) Far away animal. (f) Sunlight. (g) Night image captured with infrared camera trap. (h) Night image captured with incandescent camera trap.

3.2 Metadata

In order to help the CNN to better interpret the input images, additional metadata is added to the models. These metadata mainly concern giving the network an idea of the brightness of the input image by using a number of different parameters such as solar and lunar irradiation. Additionally, by telling the network what kind of camera is used to create the images, the network can account for differences in camera functioning. This section will discuss the different parameters used, as well as the realization of these parameters.

3.2.1 Solar irradiation

As the Sun is the largest natural source of the light on Earth, and as only a limited amount of artificial light is present in the images, the sunlight incident on Earth is an important indicator of the brightness of images. While the technology to accurately capture solar irradiance exists, no real-life irradiation measurements are available to be used for the images in the Snapshot Serengeti dataset. Because of this, the model presented by ASHRAE [59], as explained in section 2.6, will be used. This model is only an approximation of the sunlight incident on a certain location on Earth, assuming a clear sky. More specifically, a clear sky indicates a lack of clouds or shade of any other kind. Evidently, this limitation cannot guarantee an accurate value in many of the cases, yet that is not exactly the point of the model. Rather than using it for precise weather simulations, it should suffice to give the CNN an idea of the amount of sunlight present in an image. This value can be normalized, meaning only the relationship between irradiation values on different times (and at different locations) is relevant. The solar irradiation at noon for example should be higher than the solar irradiation in the early morning or the late evening. Since all image locations are in the proximity of one another, the variation in solar irradiation due to variation in location will be negligible.

General principle

Bundling the equations as provided in section 2.6, a function can be written that takes the date, time, longitude, latitude, timezone, τ_b and τ_d as inputs. Given these required inputs, the function then gives E_b and E_d as output values. Since the actual operations to perform the calculations of section 2.6 are fairly simple (the most complicated step is the calculation of an arcsine in equation 2.35), almost any programming language can be used to practically implement this function. Considering that the used software for training neural networks is written in Python, the solar irradiation calculation algorithms are implemented in Python as well. This allows for easy integration in the other code.

Location, date and time

In order to calculate the solar irradiation for an arbitrary capture event, the function described above is used. Consequently, each capture event needs to be provided with correct input values for location, date and time. The date and time parameters simply can be read from the csv data files, as they are provided in plain text. The location, required as values in longitude and latitude, poses some issues however. All images that are not blank, have their coordinates given in *Universal Transverse Mercator* UTM coordinates as x and y values. Blank images on the contrary are not provided with any coordinates at all. By taking the average x and y values of all given coordinates, an approximate value for the location of blank images is found instead. These average values are x = 712577 and y = 9722377. As converting UTM coordinates to longitude and latitude is both complex and computationally expensive [75, 76], an optimized Python library [77] is used.

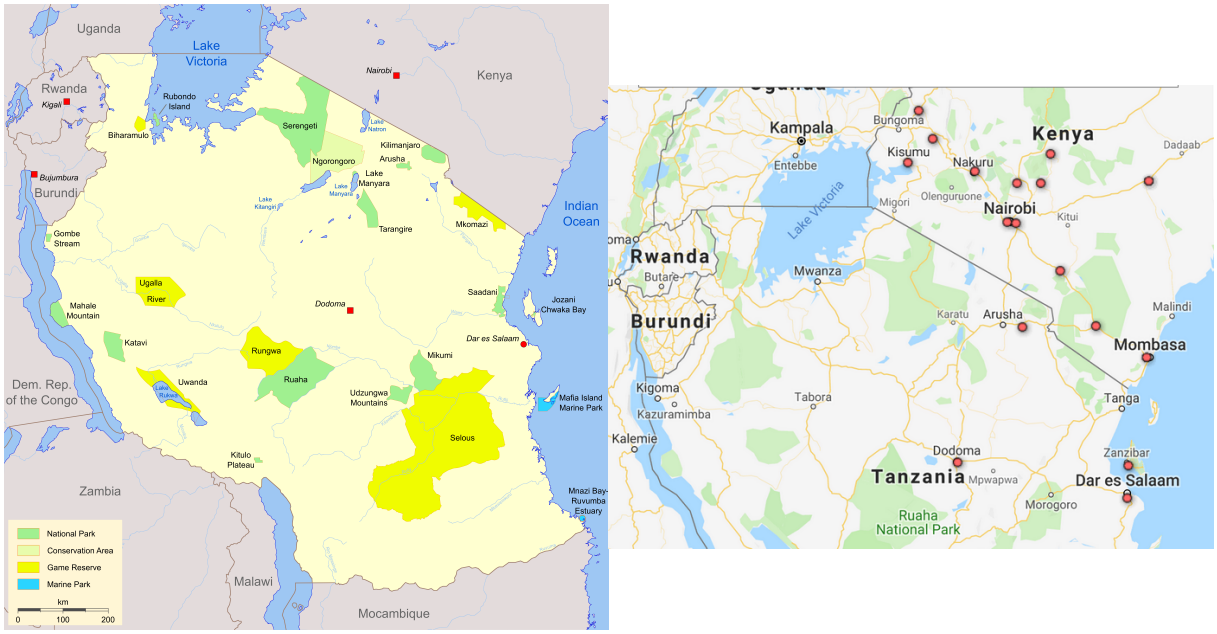


Figure 3.3: Left: Overview of the national parks in Tanzania [79]. Serengeti National Park lies to the right of Lake Victoria, near the border with Kenya. Right: Overview of the measurement stations providing meteorological parameters for ASHRAE in 2017 [78].

Location dependent parameters

The equations in section 2.6 require two location dependent parameters τ_b and τ_d as defined by ASHRAE. One way to gain access to values for those parameters is using [78]. This website provides a large number of meteorological parameters defined for measurement stations all over the world. Among other parameters, the values for τ_b and τ_d can be found. Figure 3.3 gives an overview of the stations, as well as the actual location of Serengeti National Park. While multiple stations are eligible for use in 2017, only one of those stations is also available for data from 2009 and 2013. This station, which is located in Nairobi, Kenya rather than in Tanzania, is the station that will be used in all solar irradiation calculations. In order to get the correct value for τ_b and τ_d from the tables provided in [78], two things have to be taken into account. Firstly, there is data available for 2009, 2013 and 2017. Only one of these three years can be used for a capture event. Therefore, it is necessary to determine the year that is closest to the date (year) of the capture event. The resulting closest year is the year from which the data will be used. In case of a tie, the year furthest in the past is used in this implementation. Secondly, the data presented in [78] only gives the values of τ_b and τ_d for the 21st day of each month. To get the correct value for any day, linear interpolation has to be used.

3.2.2 Lunar irradiation

The Sun may be a large source of light during the day, at nighttime no sunlight directly reaches the Earth. Instead, a fraction of the sunlight incident on the Moon is reflected to the Earth. This reflection is subject to a number of complicated processes. The positions of the Sun and the Moon relative to Earth determine the amount (flux) of sunlight reflected by the Moon to Earth, while the surface of the Moon and the inter-celestial distances determine the intensity of that light. Approximations of these processes are presented in section 2.7. It can be useful to study what the lunar influence is on pictures taken at nighttime. Therefore, by selecting a number of lunar parameters in this section and implementing those parameters as features in a CNN, their influence can be studied.

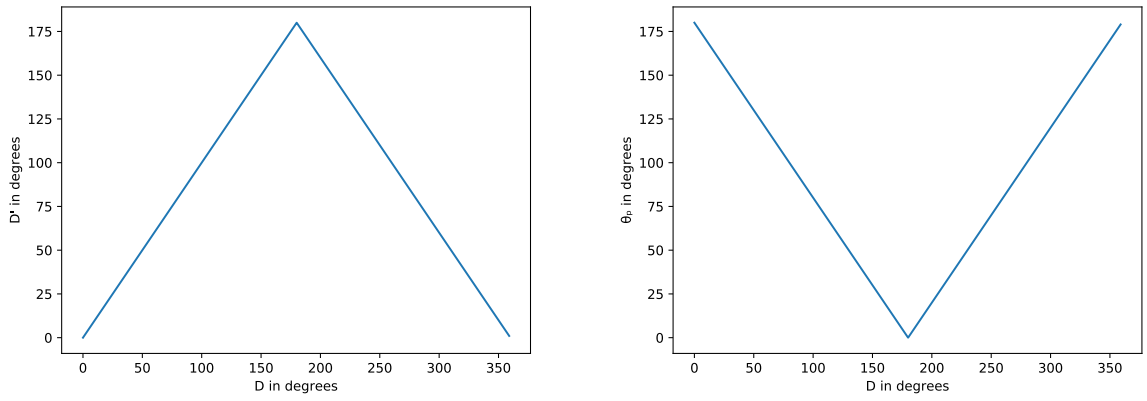
Lunar phase and lunar phase angle

Two parameters that give an idea of the amount of light reflected by the Moon are the lunar phase angle D (lunar age) and the lunar phase F. These parameters fluctuate between new Moon and full Moon and pass all phases of the Moon in between. It is common knowledge that different fractions of the Moon are lit for these different phases, meaning that the amount of moonlight also differs for these different phases. Essentially, D and F indicate the same thing, but they are expressed in different units: D in degrees and F as a fraction. Equation 2.42 gives the relationship between the two parameters. In order to be sure that any lunar influence is included, both parameters will be implemented in a CNN[2].

The procedure to calculate D and F is explained in 2.7 and can be implemented in a number of functions. Essentially, starting from a certain date and time, all necessary information is given to calculate the positions of the Sun and the Moon resulting in the phase parameters. D can then be returned as an angle between 0 and 360 degrees, while F is returned as a value between 0 and 1.

Lunar phase angle transformation

While the first part of section 2.7 describes how to calculate the lunar phase (angle), the second part describes a simple model to approximate the amount of light reaching the the top of Earth's atmosphere. This gives even more insight in the actual amount of light (energy) reaching Earth. One issue, however, is the difference in angular range between D and θ_p[3]. D starts at a new Moon at 0°, reaches full Moon at 180° and then continues to the 360° (as also given in figure 2.21). On the other hand, θ_p starts at a new Moon at 180°, reaches its full Moon at 0°and then returns to the new Moon at 180°. Its angles always lie in [0°,180°]. Figure 3.4b shows the differences. This mismatch in angles can be resolved in a couple steps given in Algorithm 2, converting D into θ_p. First, the range has to be reduced from [0°,360°] to [0°,180°] by subtracting all values of $D > 180°$ from 360° and defining the result as D'. This generates the correct range, but a full Moon in D' now corresponds to a new Moon in θ_p. To resolve that issue, the D' has to be inverted by subtracting it from 180°, resulting in the corresponding value for θ_p. The transformation is visualized in figure 3.4.



(a) D' in function of D

(b) θ_p in function of D

Figure 3.4: Visualization of the transformation of lunar age D to the lunar phase angle θ_p used in [73]. The y-axes compare D' and θ_p against D on the x-axes. Note that the ranges of both D' (left) and θ_p (right) are [0°,180°], while the range of D always is [0°,360°].

[2]As will be explained in section 3.2.7, this implementation only is in the final layer of the CNN. The implications are extensively discussed in chapter 4.

[3]D is the lunar phase angle as defined in [66], while θ_p is the same parameter defined in [73].

Algorithm 2 Lunar age D to lunar phase angle θ_p.

procedure CONVERT D TO θ_p
    if $D > 180°$ **then**
        $D' = 360° - D$
    else
        $D' = D$
    $\theta_p = 180° - D'$

Lunar irradiation

Once the the lunar phase angle has been transformed, the mean Top-Of-Atmosphere lunar irradiation $\overline{E}_{TOA}$ for that angle for a certain wavelength can be found. There is a number of issues concerning this calculation however. Firstly, the mean Sun-Moon radius $\overline{R}_{sm}$ is not provided in [73] and quite difficult to accurately calculate in general (due to the orbital movements of the Moon around the Earth and the Earth around the Sun). A rough estimate is made instead using the following equation 3.1.

$$\overline{R}_{sm} = \frac{\left(R_{se,p} - \overline{R}_{me}\right) - \left(R_{se,a} + \overline{R}_{me}\right)}{2} \tag{3.1}$$

When the Earth is in perihelion, it is on its closest position relative to the Sun. If this distance $R_{se,p}$ would be a line between the Sun and Earth, and the Moon would lie exactly on that line, the Sun-Moon distance would be the smallest possible. This distance is represented in the $\left(R_{se,p} - \overline{R}_{me}\right)$ term of equation 3.1. Similarly, if the Earth would be in aphelion (the point farthest away from the Sun) and the Moon would be positioned behind the Earth on the line through the Earth and the Sun, the Sun-Moon distance would be the largest. This distance is represented in the $\left(R_{se,a} + \overline{R}_{me}\right)$ term of equation 3.1. Taking the average of these two values gives a very rough approximate of the mean Sun-Moon distance. $\overline{R}_{sm}$ resulting from equation 3.1 is 149,596,150.3 km.
Secondly, equation 2.63 only gives $\overline{E}_{TOA}$ for one spectral wavelength. In order to have a representation for the entire visible spectrum (the light reaching Earth also spans the entire visible spectrum) a (numerical) integration operation would be necessary. As (numerical) integration is computationally expensive, it would be unwise to perform that calculation for 1.7 million images. Therefore, only the normalized results for one wavelength will be used. In this way, the inaccuracies of equation 3.1 are counteracted. That is the case since $\overline{R}_{sm}$ is a constant value. Strictly speaking, it is even unnecessary to calculate $\overline{R}_{sm}$, but it has been included in case the model would have to be used more accurately in the future (the order of magnitude is correct). The wavelength that will be used is λ_s equal to 450nm. For these values of $\overline{R}_{sm}$ and λ_s, a normalized representation of the lunar irradiation can be calculated.

3.2.3 Time

While used for other metadata (solar and lunar irradiation), the time itself can be used as a feature too. Logically, the time of the day should give an idea of the amount of sunlight present in an image. Additionally, some species may only be active at certain times of the day. Both arguments can be verified by including time itself during classification. Note that only the time of day (hour, minute, second) is included, not the actual date (day, month, year). The easiest way to include time is as a single scalar value[4], comparable to the decimal day calculated using equation 2.43. Because only hours, minutes and seconds are taken into account, the *decimal*

[4]Time consists of three values: Hours, minutes, seconds. Including all three values independently would be overly redundant. It would also complicate the neural network model to be trained for approximating this formula. Instead, these three values can be combined into one scalar value.

hour is calculated instead (equation 3.2). This is the value that is used in classification.

$$\text{decimal hour} = \text{hours} + \frac{\text{minutes}}{60} + \frac{\text{seconds}}{3600} \tag{3.2}$$

3.2.4 Camera model

In the research in [4], two different kinds of cameras are used. Initially, DLC Covert II [80] cameras were used. However, as these cameras created nighttime images of insufficient quality, the researchers have changed to Scoutguard SG565 [11] cameras, which perform better. A significant difference between these two models is that the DLC Covert II cameras capture nighttime images using an infrared flash, while the SG565 cameras use an incandescent flash (visible light). This generates two entirely different kinds of nighttime images, as demonstrated in figures 3.2g (infrared) and 3.2h (incandescent). If the network is notified of the kind of camera involved for the input image, it might compensate differently. Besides this difference, it is possible that there are other distinctions that are visible in the images. Including the camera model as metadata might allow the network to take other possible differences into account as well.
Practically, the camera model is represented as a Boolean value: A DLC Covert II gives a zero as input into the network, while the SG565 gives a one. The reasoning behind this method is that the zero shuts down certain weights in the network, while the one allows them to be used. This way, different camera models might results in completely different weights that are included.

3.2.5 NightDay

A simple parameter, *NightDay*, indicates whether it is night or day in the input image. This is done using the value of solar irradiation. Whenever that value is greater than zero, $NightDay$ is given the value one to indicate day. On the other hand, if the solar irradiation value is zero, $NightDay$ also is zero. For the same reason as in the camera model, values of zero and one may induce the use of different weights.

3.2.6 Flash and Exposure Time

Each captured image corresponds to the EXIF standard [81] and contains a number of interesting parameters. It is important to note however that these data are incorrect for all images taken with the DLC Covert II cameras. The *ShutterSpeedValue* for example is always zero, which should never be the case. That means that only the images captured using SG565 cameras contain relevant EXIF metadata. Consequently, these metadata cannot be used in classification. Otherwise, either all DLC Covert II images are unusable or all DLC Covert II images have undefined metadata input values. Despite this limitation, two metadata parameters have still been included for future study and other research (see section 3.4). Extraction of EXIF data in Python can be done using the exifread library [82].
The *Flash* parameter indicates whether the flash was fired while capturing the image. For the SG565, this parameter can take three values as shown in table 3.2[5]. Values zero and one indicate a image taken without flash, while a two indicates the presence of a flash. Studying the flash value might again provide insight in the way a CNN handles the brightness of an image.
In addition to the flash, the *ExposureTime* parameter gives the value of the exposure time in the form of values for a and b (which are simply the numerator and the denominator of a fraction). Using a and b, the exposure time T in seconds can be calculated following equation 3.3 [81]. T indicates the amount of time the camera is exposed to the light source (the environment with or without flash).

$$T = \frac{a}{b} \tag{3.3}$$

[5]This table only serves to illustrate the different values of the EXIF *Flash* parameter. When training a network using the flash, it would be unwise to use the values like this. It would be better to provide one parameter indicating whether a flash was used, and another indicating the flash mode.

Table 3.2: Different values for the *Flash* parameter in a SG565 camera.

Value	Meaning
0	Flash did not fire, auto mode
1	Flash did not fire, compulsory flash mode
2	Flash fired, auto mode

3.2.7 Implementation of metadata in a network

While accurate data acquisition already is quite elaborate, actually using the data only adds to the complexity. Since the input of any CNN is an image, the scalar metadata values cannot simply be added to that input. Perhaps methods exist to add the metadata to the the activation maps of the convolution layers in the network, but those have not been explored in this research. Instead, metadata input can also be connected to the fully connected layers at the output of the network. This is the most intuitive approach using Keras/Tensorflow, but perhaps not the most logical. While the last layers mainly perform classification, the actual image processing happens in the first layers. By only adding metadata to the last layer, an import part of image processing may not be influenced. Considering the complexity of adding metadata in other places in the network, this has not been done however.
As discussed in section 2.3.4, the final fully connected layers are the result of a (global average) pooling operation of the last convolution activation maps of the CNN. More precisely, the global averaging results in one (flattened) fully connected layer of for example 2048 hidden units (this is the case for ResNet-50 [6]). After potentially adding additional fully connected layers (which is done in for example in AlexNet [42]), the output is then fed into a softmax layer, calculating the classification score for each class. Adding metadata to the fully connected layer could then be achieved as in figure 3.5, where the metadata is concatenated directly on the flattened fully connected layer. No additional fully connected layers are added, but instead the concatenated layer is fed into the softmax layer, calculating the output. The idea here, is that the metadata features have a certain influence on the network's weights due to the backpropagation of the gradients possibly leading to a better classification accuracy. Figure 3.6 gives the alternative using additional fully connected layers instead. This way, assuming that a neural network can learn any function, an architecture that is able to beneficially use the metadata is sought. One important remark to make is that using only a few FC layers may be too little correctly model non-linearity.

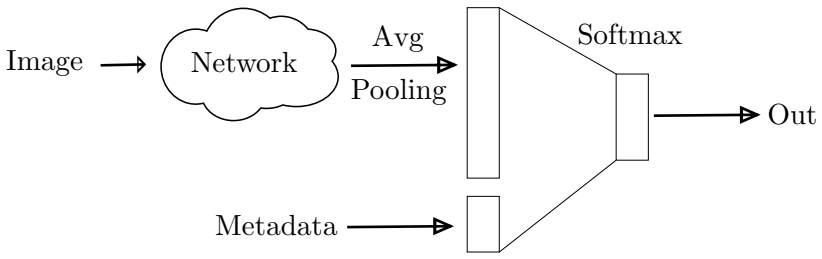


Figure 3.5: Adding metadata to the output fully connected layer without adding any additional fully connected layers. The softmax layer consists of 48 (in the case of species classification) hidden units with the softmax activation function that take the combination of flattened outputs and metadata as input.

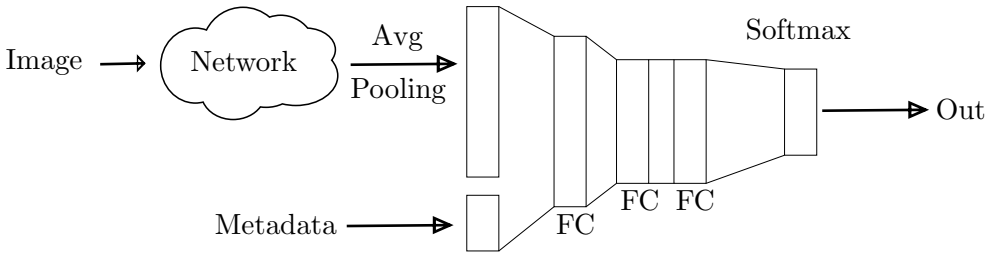


Figure 3.6: Adding metadata to the output fully connected layer while using additional fully connected layers.

3.3 Training neural networks

3.3.1 Software and hardware

Writing custom software for training neural networks is a tedious and time consuming process. Therefore, various machine learning frameworks have been developed, for example TensorFlow from Google, CNTK from Microsoft and Theano from the University of Montreal. These frameworks are full of features and allow training and interference using Graphics Processing Units or GPUs. They are not easy to comprehend however, and require a certain learning curve. That is why the framework Keras is used, which is a high-level neural network API written in Python that runs on top of TensorFlow, CNTK, or Theano. It provides a simple interface that abstracts the extensive implementations of the selected backend which is TensorFlow in this research, and allows fast and flexible prototyping [83]. The software code for this research is written in the programming language Python and is fully documented. The training and testing of neural networks is performed on two different computer systems with the specifications given in table 3.3.

Figure 3.7: TensorFlow and Keras frameworks [83, 84]

Table 3.3: Training hardware.

	Computer 1	Computer 2
OS	Windows 10 Professional	Windows 10 Professional
CPU	Intel Core i5-6600K	Intel Core i7-7700
Memory	8 GB	32 GB
Disk (for dataset)	240 GB SSD	256 GB SSD
GPU	NVIDIA GeForce GTX 1080 Ti	NVIDIA GeForce GTX 1070 Ti

3.3.2 Dataset preprocessing

The images in the Snapshot Serengeti dataset are RGB images with a resolution of 2,048 × 1,536 pixels. This resolution is too large for deep neural networks due to computational cost restrictions. Therefore, the 2,048 × 100 pixel footer is first removed and the images are downscaled to a 256 × 256 pixel resolution (see fig. 3.8). This downscaling causes the image to deform as the aspect ratio changes, although this is not an issue for neural networks.

Storing the dataset on disk under the Windows 10 operating system also requires some additional actions, as simply putting all images of one species in a single directory results in very slow file access times. Consequently, each species directory contains a number of subdirectories

that each contain 1,000 images. This simple operation causes a significant speed-up in file access. Images are named after their capture event (which has an unique string), combined with its rank in the event (the first picture would be *uniqueCaptureEventString_1.JPG*).

(a) Original 2048x1536 image (real size is not presented here).

(b) Downscaled 256x256 image. Note that the aspect ratio has changed.

Figure 3.8: Dataset preprocessing.

3.3.3 Data augmentation and normalization

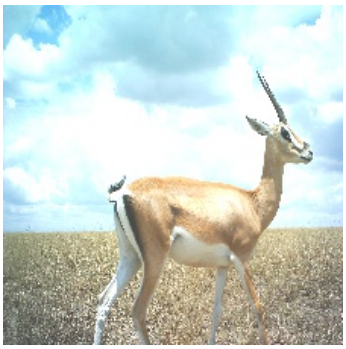
Whilst training deep neural networks, it is common practice to perform data augmentation on the input images. This makes the neural network more robust by creating slightly modified versions of the original data. Random cropping, horizontal flipping, brightness modifications and contrast modifications are implemented. Additionally, it is verified whether applying *Contrast-Limited Adaptive Histogram Equalization* CLAHE proves to be beneficial. This technique is an improvement on *Adaptive Histogram Equalization* (AHE) which is in its turn an improvement on *Histogram Equalization* (HE) [85]. HE is an image processing method wherein across the spatial domain the pixel intensities are uniformally distributed. In other words, the histogram of pixel intensities is flattened and stretched out to increase contrast. In AHE the image is first divided into blocks or tiles and then histogram equalization is performed separately on each of them. The idea is to locally enhance contrast. A problem however, are densely concentrated pixel intensities which introduce noise artifacts. The solution is applying a clip limit on the local histogram resulting in the technique CLAHE [85], which is first defined in [86]. The reason for testing CLAHE in this research is to make the animal fur pattern stand out and to improve night image quality for better classification. Note that no brightness or contrast augmentation is performed when CLAHE is used, because that would interfere with the functioning of CLAHE (that modifies brightness and contrast itself). The random cropping results in images of 224 x 224 pixels which is a standard resolution for deep neural networks. Figure 3.9 shows the image augmentation steps. Each image is also normalized by subtracting the mean and by dividing the standard deviation of the pixel values. Figure 3.10 visualizes this normalization of RGB images. The red cross points at the origin of the plot around which the normalized RGB values are centered. It should be noted that this center is not exactly the center of the original RGB values because of the specific software implementation. Therein the mean and standard deviation are calculated as the combined average of each color channel. The metadata described in 3.2 are also normalized when possible. These normalization steps improve the learning process of neural networks with faster convergence [23, 87].
One final remark that should be made is that the images are rescaled before training, but that all other preprocessing happens during training. This method is for example also used by [58]. A case could be made for performing the preprocessing on the original images, but that would be impractical in this implementation. While it might improve accuracy (as the original images contain more information), it would also introduce more delay in the training (all operations would have to be done on larger images, and every image would then have to be rescaled).

(a) Original image (256x256)

(b) Random cropping (224x224)

(c) Horizontal flipping (224x224)

(d) Decrease brightness (224x224)



(e) Increase brightness (224x224)

(f) Decrease contrast (224x224)

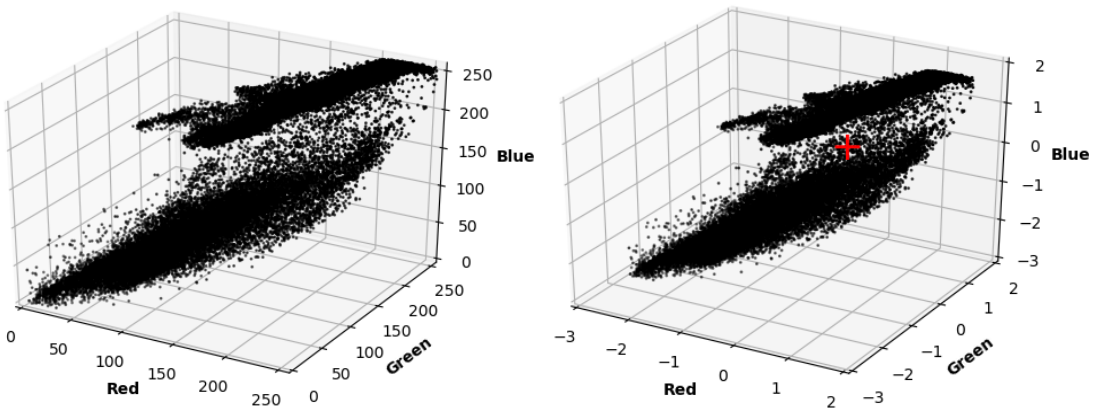
(g) Increase contrast (224x224)

(h) Contrast-limited adaptive histogram equalization (224x224)

Figure 3.9: Image augmentation.

3.3.4 Networks

Two different types of convolutional neural networks are trained, namely ResNet and MobileNetV2 (see section 2.4). The ResNet architecture has multiple variants with varying depth from which the version with 50 layers, hereby referred to as ResNet-50, is used. The Keras deep learning library has both networks available with the option of using pre-trained weights on the



(a) Original RGB values (b) Normalized RGB values

Figure 3.10: Image normalization.

ImageNet dataset. As the MobileNetV2 network is designed for mobile applications, it features only 2.22M weights (in Keras). The ResNet-50 network is a very deep network and has 23.53M weights (in Keras) [83].

There are many tunable settings whilst training neural networks, with each giving different results. The optimizer used is stochastic gradient descent (SGD) with an initial learning rate (LR) of 0.005 whilst training MobileNetV2 and 0.01 for ResNet-50, a momentum of 0.9, and a gradient clip limit of 0.01. The batch size decided upon is 50. To update the learning rate over the different epochs, a Keras feature is used that divides the learning rate by 10 in the case of MobileNetV2 and by 2 in the case of ResNet-50 once the validation accuracy has not increased by 0.0003. As appeared in early tests, there is little difference in results for weights that are randomly initialized or that are initialized using weights trained on ImageNet. Therefore, later trainings were conducted on randomly initialized weights, as is also done in [58].

3.3.5 Classification phases

Following the method of [58], classification of the images is done in two phases. In phase 1, the images are divided into two large groups, namely images containing animals and blanks. Then, in phase 2 the images containing animals are further classified into their respective species. The reasoning behind this method is that two phases help alleviate some of the class imbalance problems. As there are more blank images than all other images combined, training all classes together would be a bad idea. In such a scenario, either all blanks are used or the amount of blanks is limited. Both options are poor, as they each result in performance loss. Including all blanks would mean that the network would mainly learn to recognize all blank images. Misclassifying most of the species (except perhaps for large classes such as wildebeests or zebras) would not result in a large loss, as most of the loss generated this way would be offset by the decrease in loss due to correctly classifying the blanks. While technically still reaching high accuracies, the precision and recall (and consequently the F1 score) of the model would be poor. On the other hand, limiting the number of blanks included would decrease the amount of available data, which increases the risk of overfitting and may result in not achieving the optimal solution for the network. That is why two phases are used instead.
Note that since many different trainings have been conducted, each training will be indicated with a letter namely B for phase 1 and S for phase 2, as well as a number presented in tables 3.4 and 3.5. This will allow for convenient data presentation. Additionally, it should be noted that, unless specified otherwise, blacklisted images are used in neither training nor testing. One final note should be made regarding the difference between transfer learning, using weights that have already been trained on on another dataset e.g. ImageNet [44], and training the network without pretrained weights (training *from scratch*). [58] mentions that this difference is insignificant, and

Table 3.4: Training choices for blanks vs. species classification.

Indicator	Training settings
B1	MobileNetV2 without brightness and contrast augmentation
B2	ResNet-50 without brightness and contrast augmentation

Table 3.5: Training choices for species classification.

Indicator	Training settings
S1	MobileNetV2 with brightness and contrast augmentation
S2	ResNet-50 with brightness and contrast augmentation
S3	MobileNetV2 without brightness and contrast augmentation
S4	ResNet-50 without brightness and contrast augmentation
S5	MobileNetV2 with CLAHE
S6	MobileNetV2 with metadata features without additional FC layers
S7	MobileNetV2 with metadata features with an additional FC layer
S8	MobileNetV2 without metadata features with an additional FC layer
S9	ResNet-50 with metadata features without additional FC layers
S10	MobileNetV2, no brightness and contrast augmentation, image limit of 35,000

that they consequently only trained from scratch. While in later experiments of this research it appears that the ImageNet weights cause a slightly better Top-1 accuracy (90.38% as opposed to 88.92% as the best alternative)[6], the sentiment expressed in [58] has been followed. This not only allows for the fairest comparison, it also should give the most freedom for the weights to evolve to an optimum that uses additional metadata best.

Blanks vs. species

Unlike the classification of species (where 10 different networks were trained), the classification of blanks has been examined using fewer networks. The reasons for this are two-fold: Since the experiments for species classification were conducted first (they were deemed more important), the networks that performed the poorest could be excluded. Besides this, the training of networks on blanks vs. species classification takes a lot longer (using 643,912 blank and 520,935 species images rather than only the species images). Following the results from the species classification, blanks vs. species classification is done on MobileNetV2 (B1) and ResNet-50 (B2) using data augmentation without brightness and contrast augmentation and without additional metadata.

Species

When the images containing animals have been separated from the blanks, they can be classified into the different species present in Serengeti National Park. Considering the discussion of metadata and the use of both MobileNetV2 and ResNet-50, there are many different ways to train a CNN.
The most basic way to train a CNN is by simply presenting images at the input and evaluating the output. But even in this case, multiple training possibilities arise. More specifically, during early experiments it appeared that the brightness and contrast augmentation as presented by [58] were either too strong (removing too much information) or not of significance. That is why MobileNetV2 as well as ResNet-50 are trained in this basic scenario with (S1,S2) and without (S3,S4) brightness and contrast augmentation. Essentially the problem with brightness and

[6]It is important to note that the higher outcome for ImageNet can partially be attributed to the practical planning. Earlier in the testing phase, the acquired results for ImageNet were actually lower than those of training from scratch. Since irregularities might have happend during that earlier training, it was decided to repeat the training after all other trainings had been completed. As this new training received far more time than all others, it might have been able to climb a little higher.

contrast augmentation is that they randomly alter the lighting conditions of the images. This is unwanted whilst training with the metadata (from section 3.2) which relies on analyzing the brightness of an image. In addition, the impact of using CLAHE is examined (S5). CLAHE cannot be used for training with metadata for the same reason.
After conducting these experiments, the influence of the inclusion of metadata is then examined. This influence is examined in MobileNetV2, by concatenating features at the output (section 3.2) without (S6) or with (S7) an additional FC layer of 768 hidden units. As the amount of hidden units significantly influences the amount of parameters, 768 is the number chosen, resulting in 964K extra trainable weights[7]. This has been deemed as a good choice between allowing as much complexity as possible (more hidden units means more connections that can be found), while not entirely blowing up the network. To fully examine the influence of the FC layer, an additional CNN is trained using the same FC layer without using metadata (S8). This way, if the accuracy would raise because of adding the FC layers rather than the use of metadata, that cause can be detected. Additionally, the influence of features without new FC layers is examined on ResNet-50 as well (S9).
Finally, a basic CNN (without brightness and contrast augmentation) is trained using an image limit. No more than 35,000 images of a single class can be used (S10). Basically, this image limit allows for the examination of how to deal with the class imbalances. While this very likely will result in a lower accuracy, it might increase the performance of classifying species with a very small amount of sample images. As the image significantly decreases the amount of training data, the architecture that will be trained is MobileNetV2 rather than ResNet-50, considering that MobileNetV2 has far fewer parameters to train.

3.3.6 Evaluation

Evaluating the performance of a network is paramount to gain insight in its abilities. There are a number of values that can be calculated to evaluate the classification, and the ones that will be used here are Top-1 and Top-5 accuracy as well as F1 score.
Top-1 accuracy simply is the accuracy of correctly classifying images. If 10 images are evaluated, and 5 are classified correctly, the CNN has a Top-1 accuracy of 50%. Similarly, the Top-5 accuracy indicates whether the correct classification is among the top 5 of predicted images. More concretely, given some input image, the softmax layer of the CNN calculates a certainty score for every class. The class with the highest certainty is the predicted class, and the 5 classes with the highest certainties are the top 5.
While accuracy is a decent measure of performance, it fails to account for class imbalances [20]. That is why sometimes the F1 score is used. This score is calculated using two parameters, the precision (equation 3.4) and recall (equation 3.5) that are defined in [88].

$$\text{Precision} = \frac{tp}{tp + fp} \tag{3.4}$$

$$\text{Recall} = \frac{tp}{tp + fn} \tag{3.5}$$

In these equations, tp are the true positives (positives that are classified as positive), fp are false positives (negatives classified as positive) and fn are false negatives (positives classified as negative). The precision then can be seen as the ability of the classification to not falsely include negatives as positives, while the recall indicates the ability to correctly find all positives. For problems with more than two classes, precision and recall are calculated for each class separately by defining that class as positive and all other classes as negative. Taking an average for all classes then gives the global precision or recall. Note that for this research, these averages are weighted (which means that the classes with many samples have more influence on the average

[7] A FC layer of 1000 would result in 1,27M extra weights which is too much, while for example 500 hidden units would only result in 606K added weights.

than the classes with few samples).
Taking these measures, the F1 score can be calculated as for example in [89] and is given in equation 3.6.

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.6}$$

It is the harmonic mean of the precision and recall, and gives an idea of the network's ability to conform to both precision and recall. For multiple classes. it is calculated similarly to the precision and the recall. Once again taking a weighted average allows to find the global F1 score for classification.

3.4 Regression networks

Besides the classification of images using metadata features, it can be interesting to examine if the networks are actually able to learn to detect those metadata features. Rather than using classification to inspect the influence of metadata, this section will go into detail of how to use regression CNNs for predicting metadata values. If a CNN is able to predict the metadata value for any relevant given image, it can be assumed that the CNN recognizes the metadata features in an image. An example should put this statement into perspective. In section 3.2 one of the proposed metadata features is solar irradiation. More specifically, if the CNN knows the solar irradiation corresponding to an input image, it might be able to interpret the input image in a certain way. This way, if the solar irradiation is very high, the CNN may compensate for a high brightness in the image. To test this assumption, a regression CNN can be used to try and predict the solar irradiation in a given input image. If the CNN can indeed predict the amount of solar irradiation in an image, it is able to correctly identify indications of solar irradiation in an image. Taking this into account, if solar irradiation is provided as additional metadata in a classification task, it can be assumed that the CNN must be able to correctly identify the indications of that solar irradiation and compensate accordingly. Practically, a regression CNN is used to predict solar irradiation and exposure time.

3.4.1 Metadata to predict

There are a number of possible metadata features to predict, each of them are discussed in section 3.2. This section will briefly discuss predicting those different metadata features.

Solar irradiation is one of the features that will be trained on a regression network. Since the amount of sunlight should directly correlate with the brightness of an image, it should be quite feasible to accurately predict solar irradiation. It is important to keep in mind however that the solar irradiation used is based on a model rather than obtained through observation. This means that for example an overcast image can be correctly predicted, while the corresponding calculated solar irradiation represents a clear sky. Although this will introduce noise in the training, it should be robust enough to still present accurate results.

Where solar irradiation is clearly visible in all daylight images (since it is the prime source of light), lunar features are far less observable on nighttime images. Notably, the SG565 images that are used in training use an incandescent flash in nighttime images. This flash by far succeeds the influence of moonlight in the image. As all three lunar features (phase, age and irradiation) correspond to the amount of moonlight, it might be harder for a network to detect and correctly interpret the influence of the moonlight in an image. That is why the lunar features will not be trained in a regression network in this research.

Predicting the hour in an image, while seeming simple, is a difficult and ambiguous task. In this implementation, the network will have to use features visible in the image to predict anything. If someone is walking in nature without any device to keep track of time, they still

have two indicators of time. Firstly, the location of the sun in the sky should give a fairly accurate indication of the time. Secondly, the amount of sunlight also gives an indication of time. Of these indicators, only the second is usable in the CNN setting. While some images may show the Sun in the sky, the orientation is never given. Consequently, even if the Sun is visible (which is often not the case), it is impossible to say if it is currently in the East, the South or the West. Using the amount of sunlight (the solar irradiation) is ambiguous however, as any brightness can correspond to a time in the morning or a time in the afternoon/evening. Finally, both methods only work during the daytime. At night, when no sunlight is present, it is far more difficult to give any reasonable predication of time. All these arguments indicate it is better to not try and predict the time.

Predicting the camera model is interesting if the difference between infrared and incandescent images needs to be detected. This is classification, however, and not the focus of the research described in this section. That is why this prediction will not be trained.

Where predicting the solar irradiation seems feasible and useful, predicting the *NightDay* parameter seems a little simple. This would only imply that CNN ought to classify between light and dark images, which is no difficult problem at all. A very simple program that simply compares the average pixel intensity against a threshold would probably already achieve a good accuracy on this problem. Using deep learning would be overkill.

Similarly to *NightDay*, predicting the flash mode (as given in table 3.2) would again result in a classification. At nighttime (*NightDay*=0), 46550 of the 58358 SG565 images (79.77%) use a flash. Taking possible inaccuracies into account (images that are wrongly classified as nighttime images), it might not be useful to try and classify these images. Likewise, only 888 of 649399 daytime images (0.14%) use a flash. Classifying these images would be completely pointless, as simply always saying that no flash is used will yield an accuracy of 99.86%. Finally, since this task would be a classification task, not regression, *Flash* is not used for regression training.

ExposureTime finally is a fine target for training using regression in a CNN. Theoretically, the exposure time should be in some relation to the amount of light in the environment. As that amount of light is visible in the images, a CNN should be able to predict the exposure time of an input image.

3.4.2 Regression architecture and training

Constructing a CNN for regression rather than classification is quite simple. In principle, classification into classes only occurs in the final layer, the softmax layer. This is also the layer that needs to be modified if the network needs to train for a different number of classes. To change this from classification to regression, the only thing that needs to change is that softmax layer. By replacing the softmax layer with a single hidden unit with a linear activation, a simple regression CNN is created. Rather than creating scores between zero and one, the linear activation simply performs a weighted sum of the outputs of the previous layer. Consequently, the network output can be virtually any value, which is what regression should be.
Practically, the training occurs on MobileNetV2, exchanging the softmax layer with a single linear activation. The batch size is 50, resulting in a total of 10418 batches for solar irradiation or 8602 batches (only the SG565 images are used[8]) for exposure time. Using *Mean Absolute Error* for a loss function, the reported loss is the actual absolute error (rather than for example mean squared error). The initial LR is 0.005 with a momentum of 0.9 and a clipping value of 0.01. Whenever an epoch fails to decrease the loss with 2 W/m^2 (for solar irradiation) or $0.5s^{-1}$ (for exposure time), the LR is divided by 10. Note that rather than training using the actual

[8]This is because only the SG565 images give correct values for exposure time, while DLC Covert II values are incorrect.

value of exposure time in seconds, exposure time^{-1} is used instead. This is because the values for exposure time in seconds appear to be too small to be correctly learned by the CNN.

3.5 Summary

This chapter described the way CNNs can be used for classification and regression. First, the dataset needs to be filtered with a blacklist to remove all images that are too dubious. This filtered dataset can then be used to generate metadata features that are added to the networks via concatenation to the output of the global average pooling layer. Taking all this information, training strategies for classification and regression are defined. The next chapter discusses the results obtained using these training strategies.

Chapter 4

Results and discussion

At last, when both the theory and the methods to use that theory have been described, experiments can be conducted. This chapter gives the results for each experiment, as well as adequate discussion concerning those results. First, the calculations of the metadata (more specifically solar and lunar metadata) are examined and validated. Next, the classification of the images is discussed. This is done for both the classification of blanks as well as the classification of the species. Finally, the last section discusses the results of regression to predict solar irradiation and inverse exposure time.

4.1 Metadata calculation

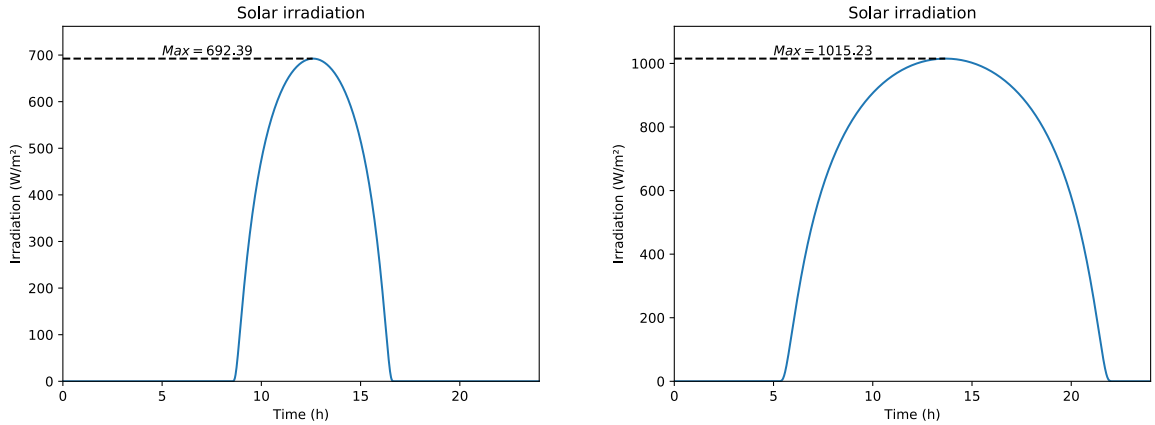
This section discusses the results of the calculation of solar irradiation and lunar features. As they are the result of modelling and calculation rather than experimentation and observation, it can prove useful to gain insight in the results they provide.

4.1.1 Solar irradiation

Calculating solar irradiation for a certain location at a certain time is utilized at multiple points in this research. Examining the results and comparing them with available real-life data gives insight in the performance of the approximation.
Figure 4.1 gives the calculated solar irradiation for the campus of Hasselt University in Diepenbeek, Belgium. With UTM coordinates of x = 668042 and y = 5644253 for zone 31N and values for τ_b and τ_d interpolated from data of [78] at Kleine Brogel in 2013 the solar irradiation at 21 June and 21 December 2011 can be calculated. While this location does not have anything to do with Serengeti National Park, it allows the calculation to demonstrate a number of facts clearly. Firstly, table 4.1 compares the times of sunrise and sunset found in [90] with the times found using the solar irradiation calculation. Sunrise is interpreted as the first time that the solar irradiation is greater than zero, while sunset is the last the time solar irradiation is greater than zero. The difference between real and calculated values is about 6 minutes, which is quite accurate. For the purposes in this research, where solar irradiation is correlated with brightness in an images, those six minutes will not make much of a difference (six minutes only account for a very small amount of sunlight).
Secondly, the calculation clearly shows how the rising of the Sun (up until midday) results in an increase in solar irradiation, while the afternoon and evening again result in a decrease in solar irradiation. These transitions demonstrate the influence of the position of the Sun in the sky.
Thirdly, comparing the maximum solar irradiation for the cases in figure 4.1, it is clear that the time of the year influences the result. Logically, the solar irradiation in the summer (June) is much higher than the solar irradiation in the winter (December). This indicates that the calculation takes season and time of the year into account. As this project does not concern Diepenbeek, however, it is necessary to check the solar irradiation for Serengeti National Park. To do this, figure 4.2 gives an overview of the average maximum daily solar irradiation per

Table 4.1: Sunrise and sunset times as found in [90] compared against the sunrise and sunset times obtained in solar irradiation calculation.

	Real Sunrise	Calculated Sunrise	Real Sunset	Calculated Sunset
21 June	05:25	05:19	21:55	22:01
21 December	08:39	08:34	16:33	16:39



(a) 21 December 2011

(b) 21 June 2011

Figure 4.1: Approximate solar irradition at Hasselt University in Diepenbeek, Belgium.

month and the average monthly solar irradiation for 2011 in Serengeti National Park. Unlike the results for Diepenbeek, in Tanzania the fluctuation is only about 100 W/m^2 (where the situations in figure 4.1 already differ in about 300 W/m^2). This graph neatly corresponds to the seasons of Tanzania [91]. On one hand, December, January and February, the hottest months in Tanzania, have the highest solar irradiation in the graph. On the other hand, June, July and August are the coldest months and have some of the lowest solar irradiation values in the same graph. This again demonstrates the functionality of the solar irradiation calculation.
To conclude, it is important to note that the normalized value of the solar irradiation will be used, rather than the absolute value. Consequently, the exact values are not important, only the relative differences and transition. As these results are realistic, the calculation of solar irradiation can be relevant for classification.

4.1.2 Lunar features

To describe and calculate lunar influence, three lunar features are defined: The lunar phase F, the lunar age D (also lunar phase angle) and the mean TOA lunar irradiation $\overline{E}_{TOA}$. This section will discuss the results of the calculations introduced in section 2.7.

Lunar phase (angle)

Discussing the lunar phase F or the lunar phase angle D basically boils down to the same thing, as they are closely correlated (equation 2.42). The main focus will therefore lie on D, as that is the parameter used in the calculation of lunar irradiation. Figure 4.3 gives F and D for June 2011. Verifying D can be done using additional data provided by [73]. In these data, the lunar phase angle θ_p (as opposed to D) is given for every hour between 1 January 2000 and 31 December 2099. Assuming that this data is correct, this is an easy way to validate the calculation of D. It is of course necessary to first transform D to θ_p using algorithm 2. Figure 4.4 gives the result of that validation for June 2011. The difference between the calculated θ_p and the θ_p provided by [73] is negligible. Only around 180° a small difference is noticeable, which consists of rounding a sharp edge of the calculated values. While not presented in any figure,

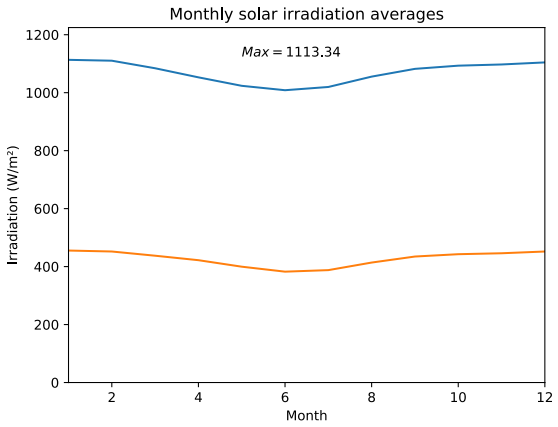


Figure 4.2: Average maximum daily solar irradiation per month for Serengeti National Park (blue). Average monthly solar irradiation for Serengeti National Park (orange). Note that there is little fluctuation in both values.

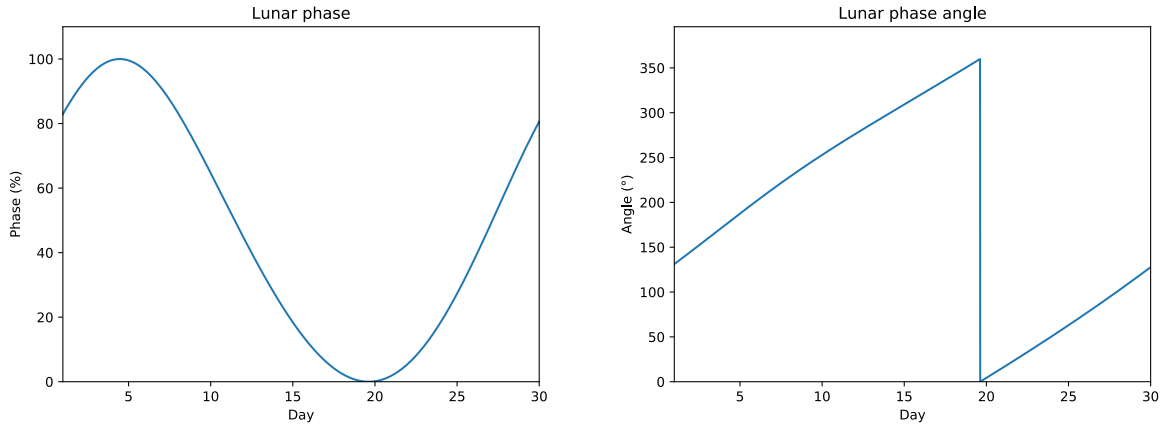


Figure 4.3: F (left) and D (right) for June 2011. Note that while D appears to be linear (with a discontinuity at 360°), this is not entirely the case.

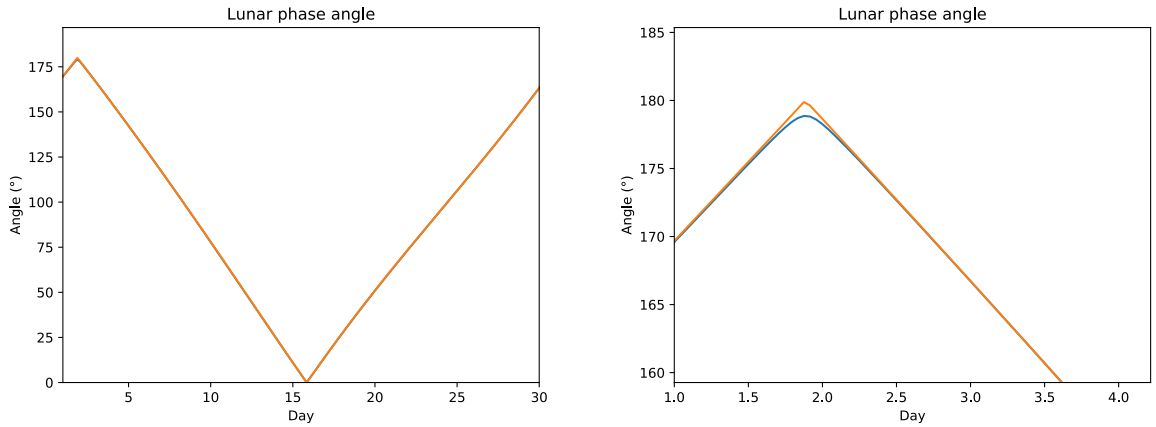
other months report the same result. It can be concluded that the lunar phase angle D, and consequently the lunar phase F, are correctly calculated. They provide an accurate enough representation of reality. Accordingly, D and F should provide insight in the moonlight present at night.

Lunar irradiation

While D and F already provide some insight in the moonlight present at night (or rather, on the portion of the moon that is illuminated), the amount of light reflected on the moon and incident on the Earth is affected by a number of complex parameters (such as the craters on the surface of the moon and the inter celestial distances [73]). Rather than going too deep into detail concerning these details, the resulting calculated values for $\overline{E}_{TOA}$ will be discussed.
Figure 4.5 gives the approximate mean TOA lunar irradiation from 1 June 2011 at 22:00 to 30 June 2011 23:00 for λ_s=450nm (this specific range has been chosen because it gave the clearest representation). Once again, the resulting graphs periodically repeat themselves every month[1], meaning that discussing one period satisfies to discuss the entire range of time (regarding the Snapshot Serengeti project).

From the results in figure 4.5, it is immediately clear that the amount of light reflected on

[1]Actually, this is a little less than a month. The Moon cycle repeats itself every 29.53 days according to [67].



(a) June 2011.

(b) Close-up.

Figure 4.4: Validation of the lunar phase D angle by comparing its transformed θ_p (orange) to data provided in [73] (blue). The close-up on the right shows the only difference in the comparison. As everywhere else the calculated and provided θ_p are the same, the blue graph is not even visible on the left (completely covered by the orange graph).

the surface of the moon is not linearly correlated with the lunar phase angle. Consequently, only when a large margin of the Moon's surface is illuminated will the reflected light be significant: The $\overline{E}_{TOA}$ is high whenever the lunar phase is near a full Moon, but very small or negligible in all other cases. Therefore, this peak indicates that if moonlight influences the brightness at night, it will be around a full Moon. Note that while the absolute value of the peak is very small (about 0.0025 W/m^2), this is only the $\overline{E}_{TOA}$ for a wavelength of $\lambda_s = 450$nm. Calculating $\overline{E}_{TOA}$ over the entire visible spectrum would result in a higher value (although that still would not matter, as the values are normalized during training). Interestingly, the model described takes into account infrared radiation[2] (wavelengths longer than those of visible red light) as well as visible light, meaning that the obtained results also count for the infrared images taken with the DLC Covert II cameras. Concluding whether this resulting lunar irradiation is influential or useful in classification however, will be done in the discussion of the classification results.

4.2 Classification

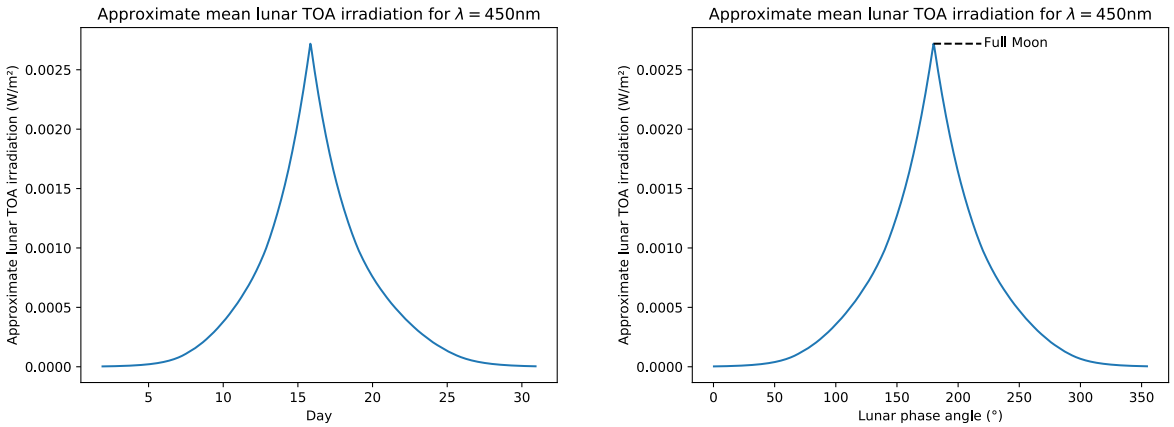
This section presents and discusses the results of both convolutional neural networks MobileNetV2 and ResNet-50 after training on the Snapshot Serengeti dataset for both classification of blank images and animal species images.

4.2.1 Blanks vs. species

Table 4.2 gives the resulting test accuracies for the classification of blanks on MobileNetV2 and ResNet-50. Additionally, tables 4.3 and 4.4 give the precision, recall and F1 score for the classification of blanks on MobileNetV2 and ResNet-50.

Immediately, it is clear that both MobileNetV2 as well as ResNet-50 reach very high classification scores. Because the classes are balanced, the accuracies and F1 scores even are nearly the same for each architecture. Moreover, these results are even higher than the accuracies reported in [58]. Where Norouzzadeh et al. reported a maximum classification accuracy of 96.8% for the VGGNet architecture (which means all other accuracies were lower), both MobileNetV2 and ResNet-50 are slightly better. Of course, with differences of only 0.13% (MobileNetV2) and

[2]This can be seen in table 2.4, where wavelenghts of up to 1.20μm are included in the calculation. It is important to remember that Si-based cameras can only capture light with wavelengths of up to about 1μm [92].



(a) (b)

Figure 4.5: Results for the calculation of the approximate mean lunar irradiation $\overline{E}_{TOA}$. On the left $\overline{E}_{TOA}$ is given in function of the day, on the right it is given in function of the lunar phase angle D. Both graphs are calculated for the range of 1 June 2011 at 22:00 to 30 June 2011 23:00. Note that the lunar irradiation is maximum at full Moon (180°).

0.53% (ResNet-50[3]), it is hard to say whether the classification is significantly better. As it is very hard to improve at all at very high accuracies, even very small differences have to be taken into account. One reason for these results it the use of a blacklist (as described in section 3.1). By removing very dubious images from the classification process, the networks more correctly learn exactly what are and what are not blank images. Interestingly, if the blacklisted images are included in the test set, the obtained accuracy for ResNet-50 is 96.45% which is a decrease of about 0.9%. As the number of blacklisted images is only about 9,000, it appears that they have a large influence. While this score is slightly lower than the maximum of [58], it is still higher than the accuracy obtained by ResNet-50 in that research (96.3%). It can be concluded that the classification of blanks is very effective, but also that images that are hard to classify for humans are also hard to classify for CNNs.

Table 4.2: Blanks classification accuracies for MobileNetV2 and ResNet-50. Norouzzadeh is added for comparison.

Indicator	Architecture	Accuracy (%)
B1	MobileNetV2	96.93%
B2	ResNet-50	97.33%
Norouzzadeh	VGGNet	96.8%
Norouzzadeh	ResNet-50	96.3%

Table 4.3: Classification of blanks on MobileNetV2.

Class	Precision (%)	Recall (%)	F1 score (%)	Number of images
Blank	96.56	97.93	97.24	137960
Not blank	97.40	95.70	96.55	111890
Weighted average	96.94	96.93	96.93	249850

[3]ResNet-50 only obtained 96.3% in [58], this comparison is against the result of VGGNet.

Table 4.4: Classification of blanks on ResNet-50.

Class	Precision (%)	Recall (%)	F1 score (%)	Number of images
Blank	96.65	98.59	97.61	137960
Not blank	98.22	95.78	96.98	111890
Weighted average	97.35	97.33	97.33	249850

4.2.2 Species

Figure 4.6 gives the resulting Top-1 and Top-5 accuracies, as well as the F1 scores for all 10 networks trained for species classification. Additionally, table 4.5 gives the same results for more easily available details. In this table, the precision and recall are included as well. Finally, in appendix A.1 complete tables are included, giving precision, recall and F1 score for each class in every network. This way, the results can be studied even more in-depth.

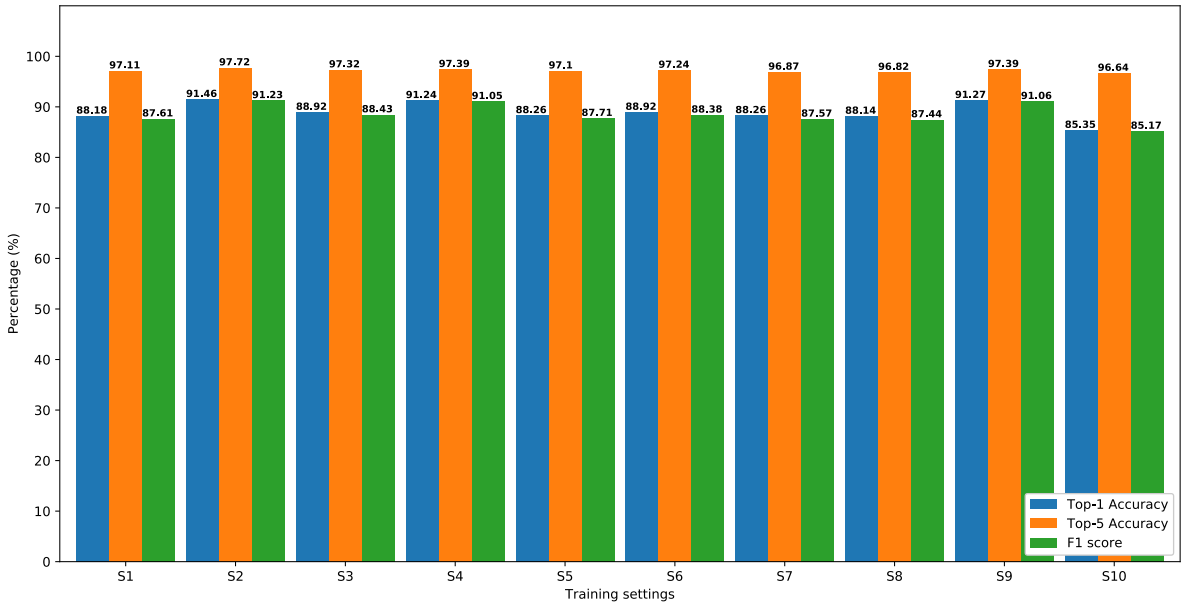


Figure 4.6: Results of each network trained on species classification. Note that the three networks crossing the 90% for their Top-1 accuracies are the three networks with the ResNet-50 architecture.

One of the first observations when studying the results is that the three ResNet-50 networks consistently have a better performance than all other MobileNetV2 networks. This confirms the idea that deeper neural networks generally should result in better performance (when paired with sufficient training data). The differences between the ResNet-50 results are very small or perhaps even negligible across all values. For the MobileNetV2 results however, there is a noticeable difference. More specifically, the S3 and S6 networks are the best, with around 0.7% difference with the other MobileNetV2 architectures. These are the networks that did not use brightness and contrast augmentation (as S1 is the only MobileNetV2 network that used those). S6 did use metadata though, which is concatenated to the global average pooling output data of the network. There is a number of conclusions that can be drawn from these observations.

Firstly, for MobileNetV2, it appears that brightness and contrast augmentation provide no added value to the training. This can be seen as the accuracy and F1 score of S3 (without brightness and contrast augmentation) are higher than those of S1 (with brightness and contrast augmentation). Moreover, it could be argued that they actually decrease the accuracy of the network, which should be avoided. For ResNet-50, the accuracy and F1 score of S2 (with brightness and contrast augmentation) are actually about 0.2% higher than for their counterparts

Table 4.5: Species classification results.

Setting	Table	Top-1 acc (%)	Top-5 acc (%)	Precision (%)	Recall (%)	F1 score (%)
S1	A.2	88.18	97.11	87.54	88.18	87.61
S2	A.3	91.46	97.72	91.15	91.46	91.23
S3	A.4	88.92	97.32	88.26	88.92	88.43
S4	A.5	91.24	97.39	90.93	91.24	91.05
S5	A.6	88.26	97.10	87.54	88.26	87.71
S6	A.7	88.92	97.24	88.26	88.92	88.38
S7	A.8	88.26	96.87	87.30	88.26	87.57
S8	A.9	88.14	96.82	87.08	88.14	87.44
S9	A.10	91.27	97.39	90.94	91.27	91.06
S10	A.11	85.35	96.64	85.37	85.35	85.17

in S4 (without brightness and contrast augmentation). As these differences are so minute, it is dubious to say that brightness and contrast augmentation actually help at all in these scenarios. At any rate, brightness and contrast augmentation seem to be unnecessary or even detrimental in training a CNN on the Snapshot Serengeti dataset. There are a number of possible reasons for this. One could be that, due to the very large amount of training images, the new images added by brightness and contrast augmentation are unnecessary. Furthermore, it is possible that the adaptions created by these augmentation methods are adapted in such a way that they actually are no longer representative of the real dataset. This would imply that the augmentations need to at least be tuned down.

Secondly, using metadata as additional features in the network does not seem to improve classification at all. Simply concatenating that data to the output of the global average pooling produces nearly exactly the same results compared to when no metadata is used. This can be observed when comparing results between S3 and S6 (MobileNetV2) or between S4 and S9 (ResNet-50). Most noteworthy is that adding FC layers does not improve the performance of the network with or without additional metadata. Rather than increasing the accuracy, this decreases the accuracy instead. One reason for this might be that both used architectures (MobileNetV2 and ResNet-50) use global average pooling to transform their final activation maps into hidden units. As argued in [51], these are meant to be directly mapped to the softmax layer. Adding additional FC layers might interfere in that philosophy, actually reducing the effectiveness of that global average pooling. As neither adding FC layers nor simple concatenation using metadata improves classification whatsoever, metadata appears to not help at all. There are four possible reasons for this. One reason would be that there is no correlation between the images and the used metadata, thus no improvements can be obtained using metadata. This is not the case however, as is proven in section 4.3 where a correlation between the images and solar irradiation (or exposure time, which has not been used for classification) is found. A second reason could be that rather than adding the metadata at the end, it should be added at the input or as some operation in the activation maps. As it is not trivial to do this in Keras, this possibility has not been covered in this research. Thirdly, it is possible that the metadata does not influence classification because the network does not need it. If it can for example adjust according to the image brightness, without needing to be told the corresponding solar irradiation, then these metadata values are indeed unnecessary. Finally, it is possible that another DNN is necessary to correctly combine the metadata with the flattened outputs, because new non-linear relations have to learned. That would imply far more FC layers are necessary than that are used now. This could be an useful possibility to explore in the future.

Thirdly, the Top-5 accuracy seems to be somewhat redundant, as over the entire range of networks the greatest difference is only 1.08% (between S2 and S10). This can be attributed to the fact that the high Top-1 accuracies guarantee the correct prediction to be nearly always in the top 5. The few times the top 1 prediction is incorrect, the probability that one of the other

images in the top 5 is correct instead is very large. Consequently, the Top-5 accuracy is not used much in the discussion of other observations, as it would add nearly no new information.

Fourthly, these results can be compared against the results that are obtained in [58] as also shown in table 2.1. MobileNetV2 has not been used in that research, but the results obtained on ResNet-50 can be used for comparison still. Norouzzadeh et al. obtained a Top-1 accuracy of 93.6% and a Top-5 accuracy of 98.4% on the same dataset. The Top-1 accuracy is about 2% higher than that of S2 (which obtained the best results) while the Top-5 accuracy is only 0.7% higher[4]. This is the case for all networks trained by [58] (except for perhaps NiN, AlexNet and VGGNet that each only obtained about 1% better accuracies). The differences between [58] and this research are therefore quite small. It is hard to say whether they are caused by some operation in this research or by an undocumented operation in the training of [58]. Many of the design choices in this research are inspired by their reported design choices. Moreover, it is also hard to say whether their reported results are accurate and correct, as they for example do not use a cross-validation set. That also increases the number of available training data, which could also play a role. Moreover, as the compositions of their training and test sets are unknown, it is impossible to make an exact comparison. Due to being unable to practically use the code they provided along with their results, it is also impossible to say whether the differences are caused by some difference in the dataset used (which should not be present). Nonetheless, the resulting accuracies of these projects are very similar and very high, which is the main conclusion that should be drawn.

Fifthly, comparing the performance of ResNet-50 against the performance of MobileNetV2, it appears that the best results of the latter are only 2.54% lower than those of the first. Taking into account that ResNet-50 has 23.53M parameters where MobileNetV2 only has 2.28M parameters, the difference is remarkable. It appears that MobileNetV2 can almost equal the performance of ResNet-50 with only a tenth of the parameters. This implies that embedded solutions could exist for this problem, using as little parameters as possible while retaining a good performance. It must however be noted that, as discussed below, MobileNetV2 fails to classify some of the classes with a very small presence in the training set. The goal of the embedded solution would have to determine whether this weakness is of relevance.

Finally, the class imbalance and image limiting should be discussed. As can be observed in for example tables A.2, A.3, A.4 or A.5, the very small classes are not classified. In table A.3 it can be seen that classes such as Aardwolf, Civet or Genet have a precision, recall and F1 score of 0%. This is because the number of images (the Civet has 10 test images) is so small that it is nearly impossible for the CNN to learn to recognize these species. It is very important to note however that ResNet-50 appears to be much better at recognizing these small classes than MobileNetV2. One example of this is that both S1 and S3, the two best performing networks with the MobileNetV2 architecture, fail to recognize the Bushbuck (which has 42 test images) at all. S2 however does manage to recogize the majority of these images, achieving a F1 score of 59.26%. Still, there are some classes that are not recognized at all for ResNet-50 too. To try and resolve that issue, S10 is trained with an image limit of 35,000 images per class, similar to an method proposed by [58]. Because MobileNetV2 has far fewer parameters to train, thus requiring less training data, this image limit is implemented on MobileNetV2. Logically, this results in a lower accuracy (Top-1 and Top-5) and also in a lower F1 score as that score is weighted. The classes that suffer most under the limit of 35,000 images are also the classes that have the most weight in the calculation of the F1 score. When comparing the amount of classes with a F1 score of 0%, no difference is observed however (both have 12 classes with 0%). One final way to check for a difference is comparing the non-weighted average (simply the sum of all F1 scores divided by 48) for both trainings. S3 in this case has an average of 47.76%, while

[4]This is again because the major classes are in the top 5 predictions most of the time. Since the 3 largest classes make out 63% by themselves, it can be accurate to say to that 63% of the images are nearly guaranteed to have a correct top 5.

S10 only has 45.45%[5]. That would indicate that, even if some small classes have increased in F1 score (which is the case with for example the Bat-eared fox), the decrease in F1 score for the classes that lost many images is greater. To conclude, it appears that using an image limit of 35,000 is not a good solution for class imbalance.

4.2.3 Classification examples

Analyzing and discussing the results from the trained networks on blank and species classification provides insight about their accuracy to correctly classify images. The problem however, is that while an image might be correctly classified (e.g. a zebra), it is unknown how sure the network is about its prediction (e.g. 20% vs. 90%). Therefore, classifications need to be visualized with the image and corresponding predictions. The following figures are generated using the best trained networks, ResNet-50 B2 for blanks classification (see tables 3.4 and 4.2) and ResNet-50 S2 for species classification (see tables 3.5 and 4.5).

Figure 4.9 shows examples of blanks vs. species classifications. Remarkably, these (correct) predictions have certainties of about 99%. Moreover, 93.52% of the test set images are predicted correctly if the network has a confidence of at least 90% as can be seen in figure 4.7. Note that the confidence threshold in this context signifies that only images with a confidence greater than the threshold are counted as being classified correct. Images that are classified correctly but have a confidence smaller than the threshold are still counted as being classified incorrectly.
Next, figures 4.10 and 4.11 show examples of species classifications. In these figures, the correct prediction accuracy is around 99%. 85.65% of all species images in the test set are predicted correctly if the network is more than 90% certain following the information in figure 4.8.
Of course, there are also cases where the network fails to predict the target class. Figure 4.12 shows such errors for blanks vs. species classifications. The most prevalent issues for wrong predictions are vegetation and rocks that appear to be an animal to the network as well as mislabeled images. However, sometimes the network simply fails without clear indications as to why that is the case.
Figure 4.13 shows errors during species classification. Some of the problems for example are animal species that have similar shapes, fur patterns and colors, or images that contain partially visible animals. Often, these problems are combined. For example, impala, Thomson's gazelle and Grant's gazelle are quite similar in shape which may confuse the network. Again, the network sometimes simply fails without any apparent reason.

To visualize what the network is looking at (is activated by), heatmaps can be drawed over the input image. The principle of generating such heatmaps or class activation maps (CAM) is described in [93]. In short, the activation maps of the last convolution layer in the network (in this case ResNet-50) are weighted by the weights connecting the global average pooling layer with the target output class neuron in the softmax FC layer and are then summed. The resulting CAM is then rescaled and overlayed on the input image with the color red indicating a strong activation whilst the color blue signifies no activation. Figure 4.14 shows some examples of heatmaps wherein it is clear that the netwerk is activated by the animal(s) present in the images.

[5]These low percentages are caused by the presence of 0% classes. As this average does not take into account the number of images per class, these classes suddenly have a lot more influence on the average.

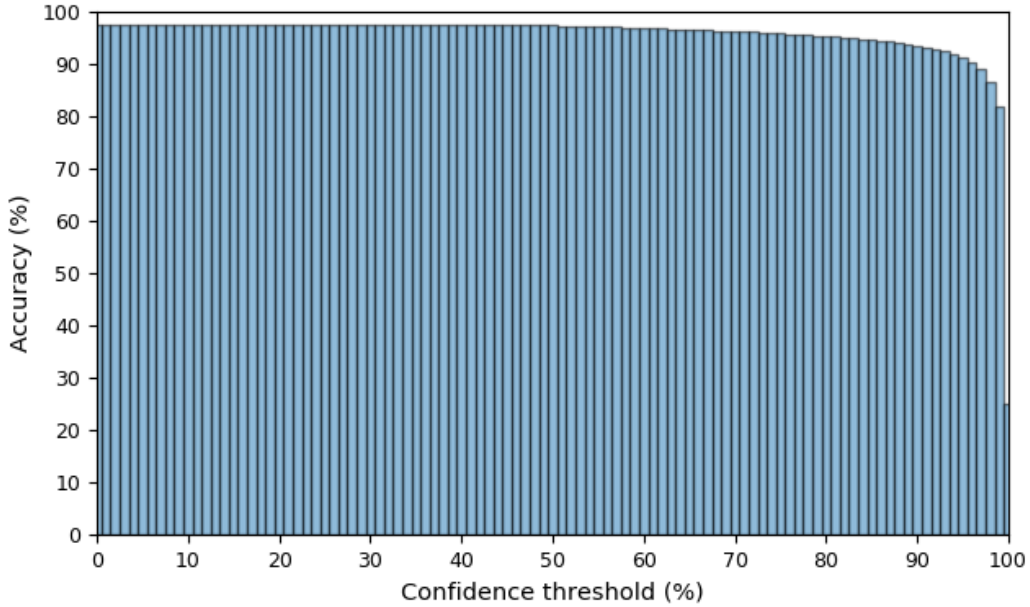


Figure 4.7: Top-1 accuracy in function of the confidence for blanks vs. species classification. If the confidence needs to be higher, the top-1 accuracy will decrease.

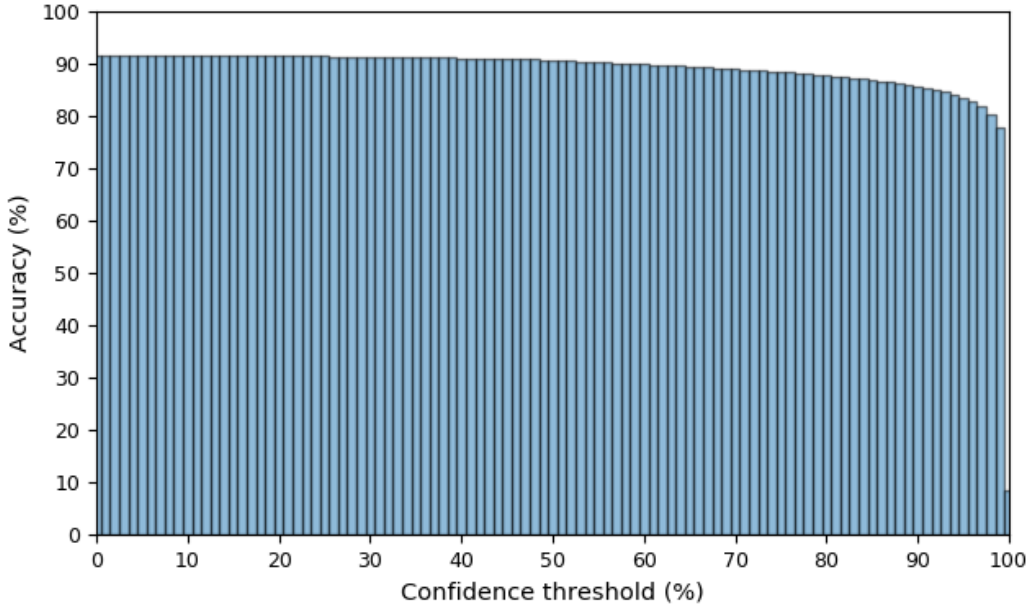


Figure 4.8: Top-1 accuracy in function of the confidence for classification of individual species. If the confidence needs to be higher, the top-1 accuracy will decrease.

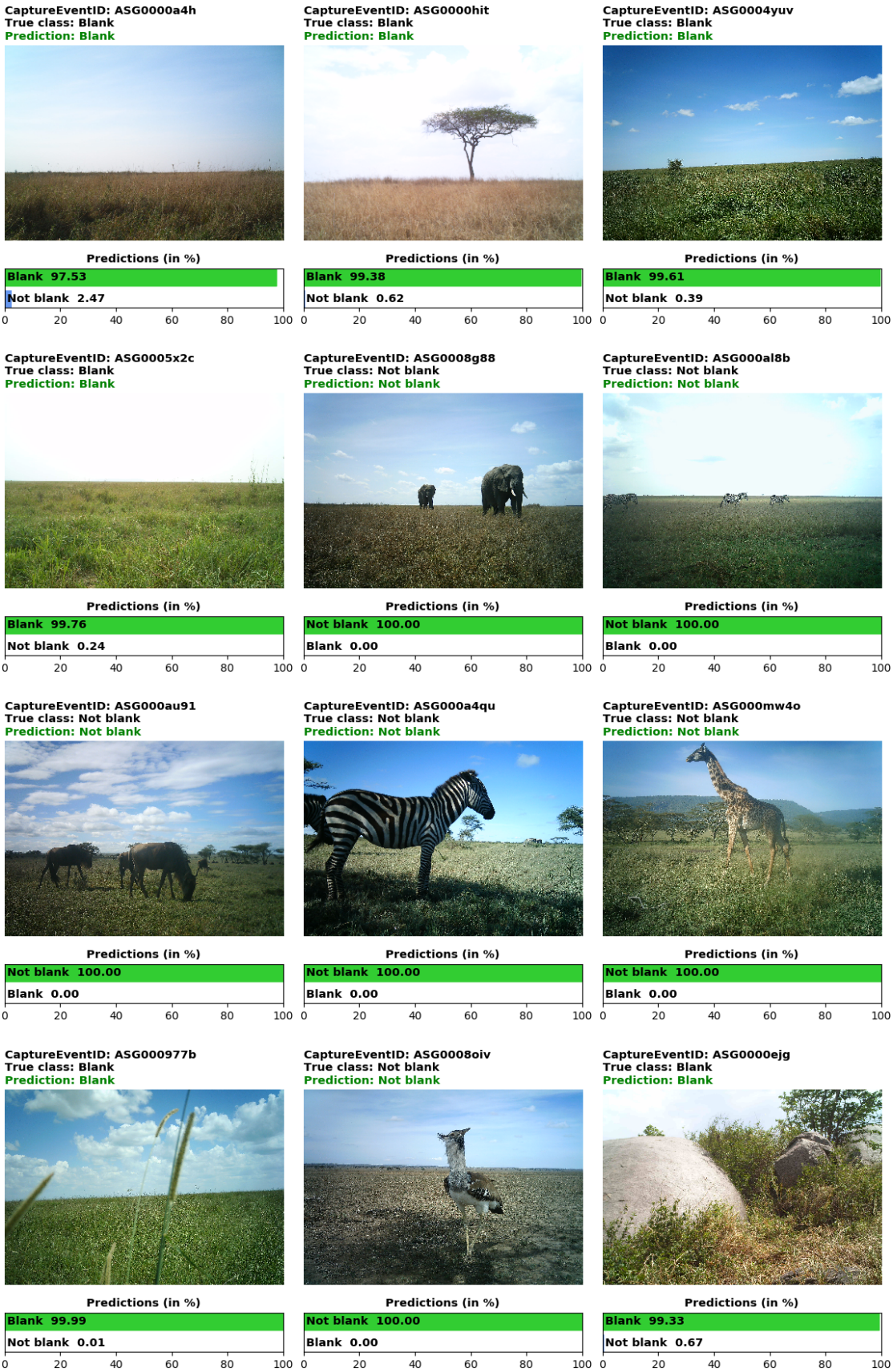


Figure 4.9: Blanks vs. species classifications.

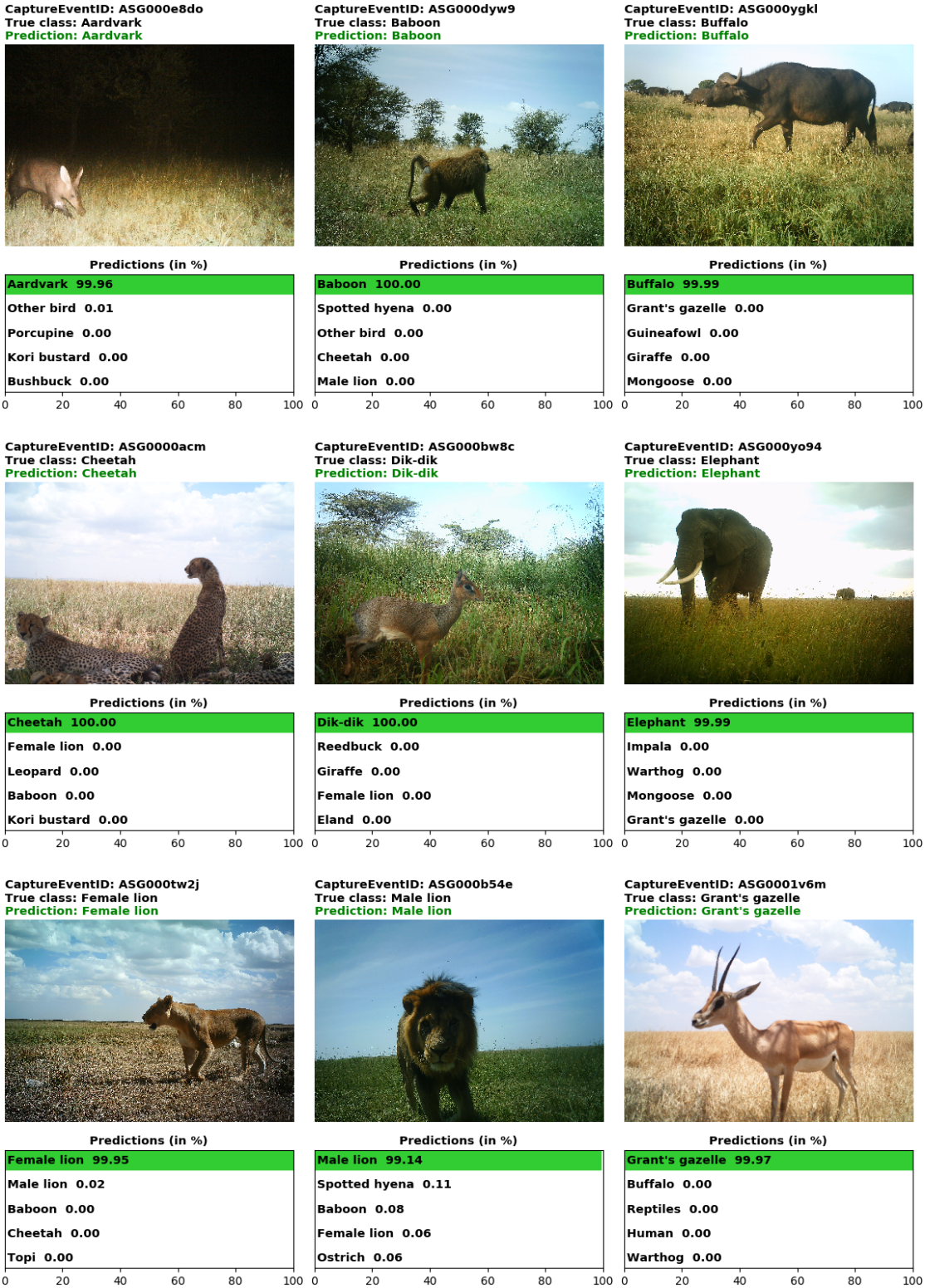


Figure 4.10: Species classifications (part 1).

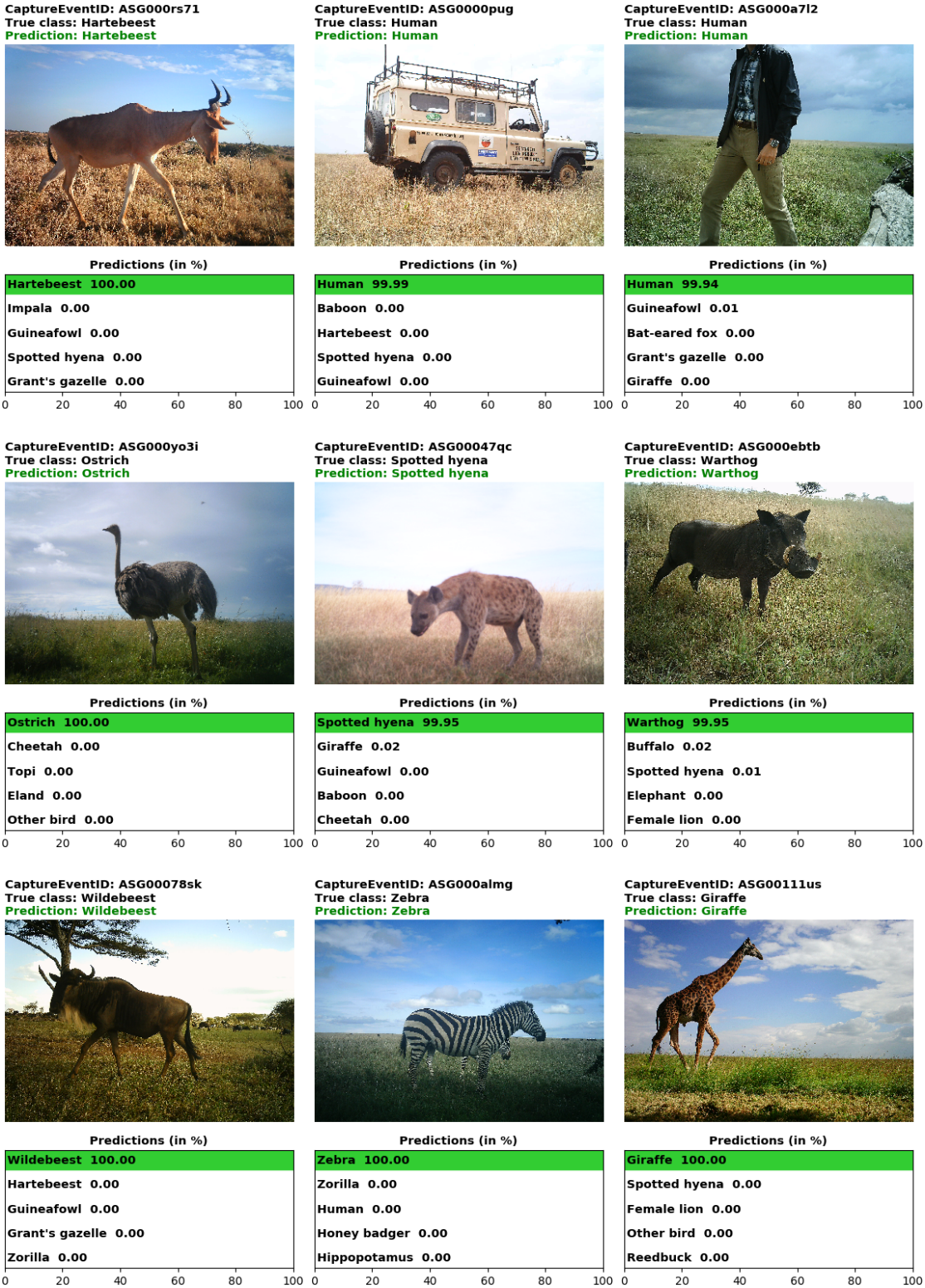


Figure 4.11: Species classifications (part 2).

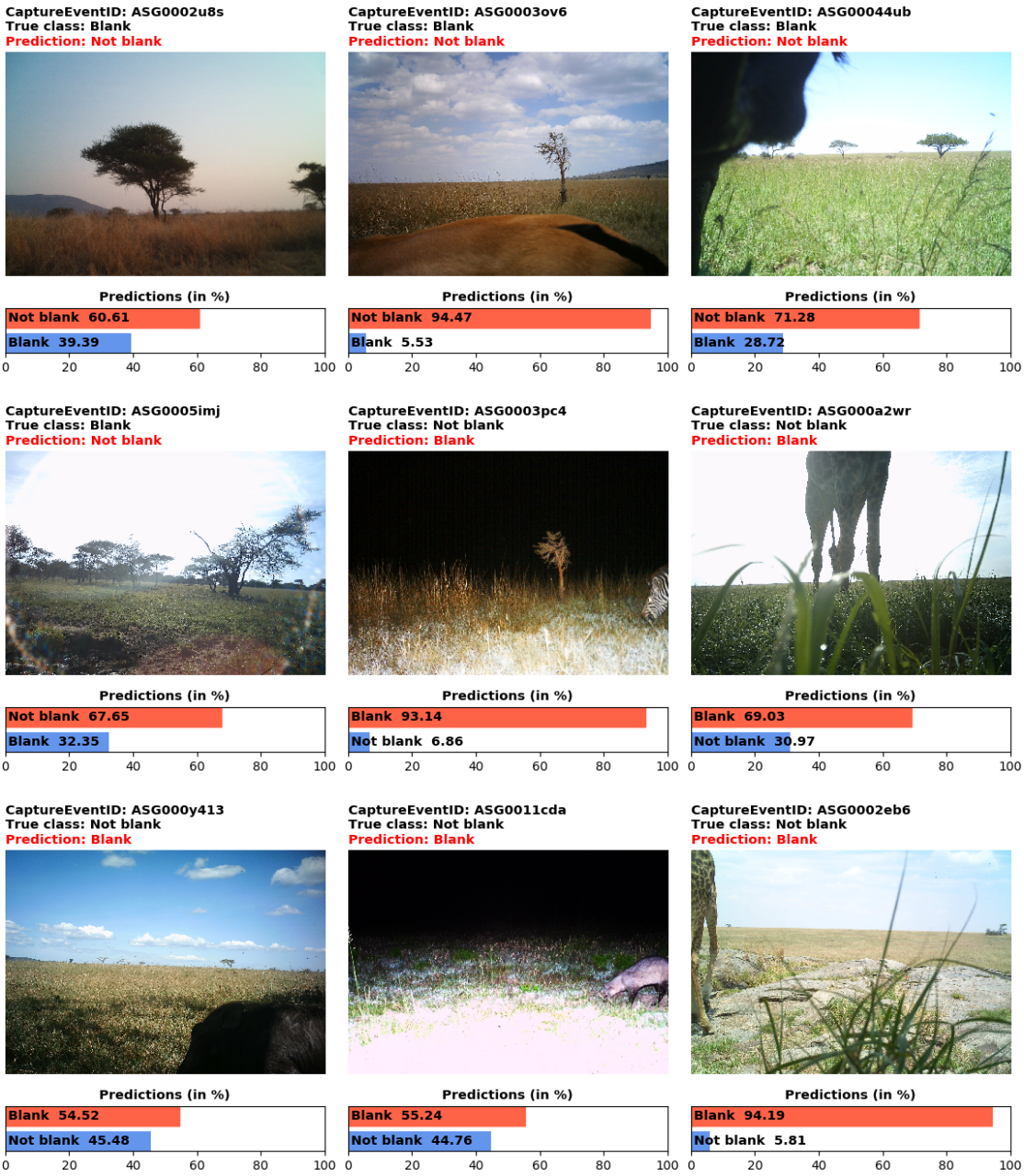


Figure 4.12: Wrong blanks vs. species classifications.

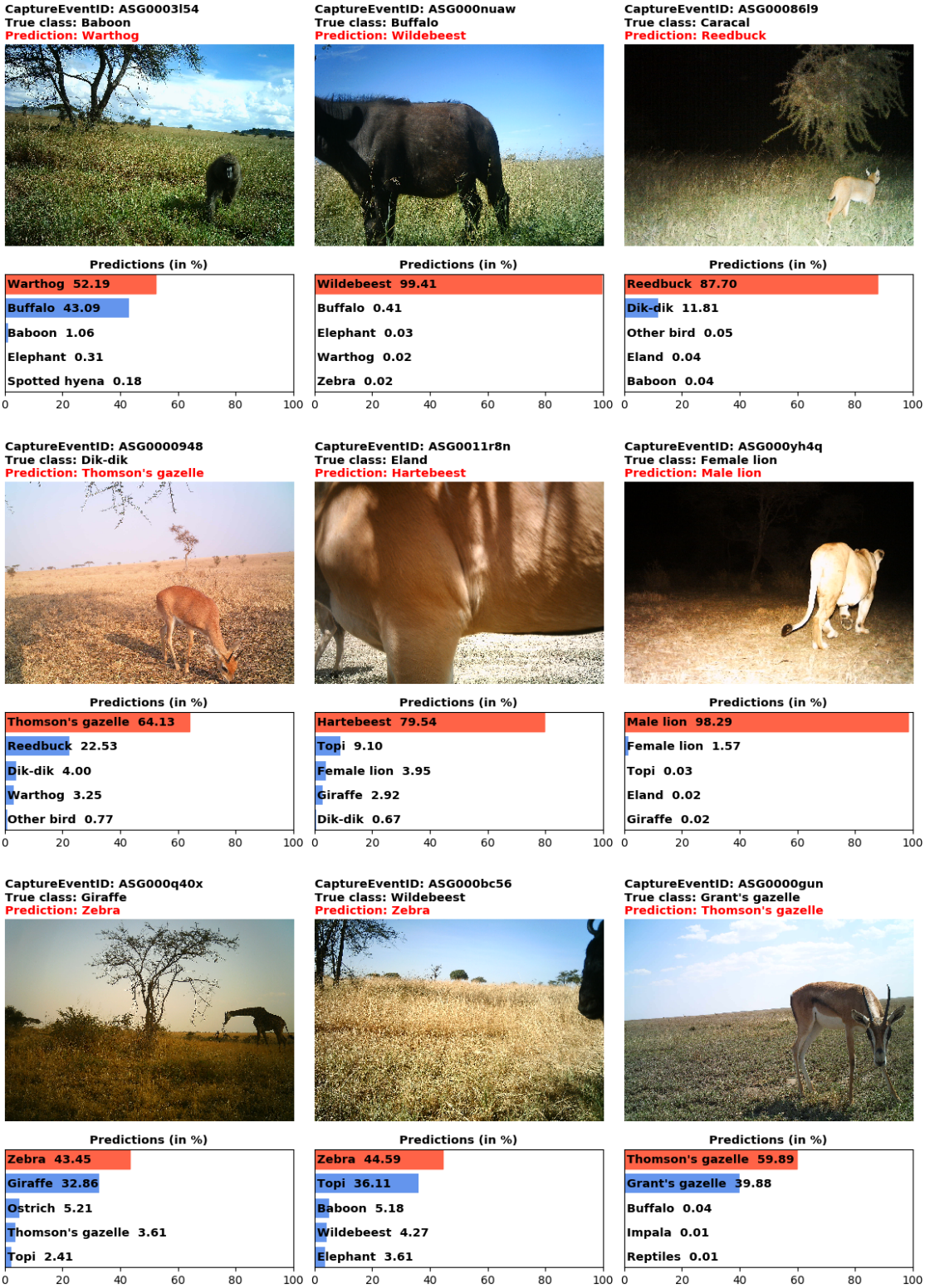


Figure 4.13: Wrong species classifications.

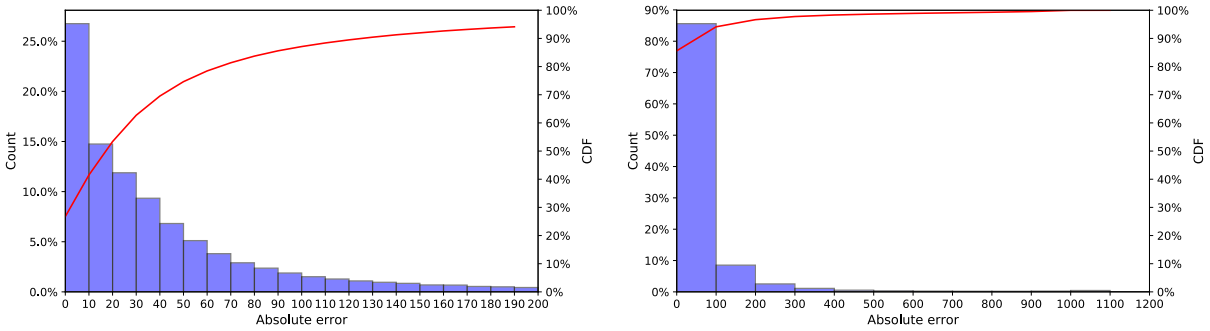
Figure 4.14: Heatmaps

4.3 Regression

Experiments were conducted to see if a CNN could learn to recognize features such as solar irradiation or camera exposure time using only an image as input data. This section gives the results obtained and discusses their implications.

4.3.1 Solar irradiation

Where accuracy, precision, recall and F1 score are useful tools to evaluate the performance of a classification network, they are useless when evaluating a regression network. Similarly, simply displaying the average absolute error for the CNN will not provide much insight either, as that error can be caused by many small (but significant) or a few larger errors. Instead, figure 4.15 gives two histograms and Cumulative Distribution Functions CDF that provide insight in the distribution of the error. Additionally, table 4.6 gives a summary of the CDF of the absolute error.



(a) Absolute error up to 200 W/m^2.

(b) Entire range of absolute errors.

Figure 4.15: Histogram and CDF of the absolute error of the test results of the evaluation of the solar irradiation regression CNN. On the left a detailed view of the absolute errors up to 200 W/m^2, while the left provides an overview of the entire error range. Note that the vast majority of the errors occurs in the first 40 W/m^2, while very large errors occur as well.

Table 4.6: Summary of the Cumulative Distribution Function of the absolute test errors using the CNN regression network for predicting solar irradiation.

Percentages	10%	20%	30%	40%	50%	60%	70%	80%	90%
Error (W/m^2)	0.07	5.32	10.91	17.14	24.46	33.02	44.29	61.61	95.03

In the CDF, it is clear that 50% of the test images causes an absolute prediction error of less than 24.46 W/m^2. For an average monthly maximum of over 1000 W/m^2, that is an error of about 2%. 70% of the testing samples have an absolute error of less than 44.29 W/m^2, and for 90% of the testing samples the error still is less than 95.03 W/m^2. These results give the impression that the predictions of the solar irradiation are fairly accurate, and often only result in a small error. However, a histogram or a CDF still is very global, not very specific.

To provide more insight in the actual predictions that are performed, figure 4.16 gives four graphs that plot the calculated solar irradiation for a specific date. In addition, they place a red point for every test image that was captured that day. This allows for a nice observation of behaviour of the predictions compared to the calculated values. The general trend for the points is to neatly follow the shape of the calculated results, often with a small error. As the calculated values have not been validated using local solar irradiation data, it is impossible to state that these small variations are erroneous. It is equally likely that the calculated values are slightly off and that the predicted values are closer to the actual solar irradiation. Some

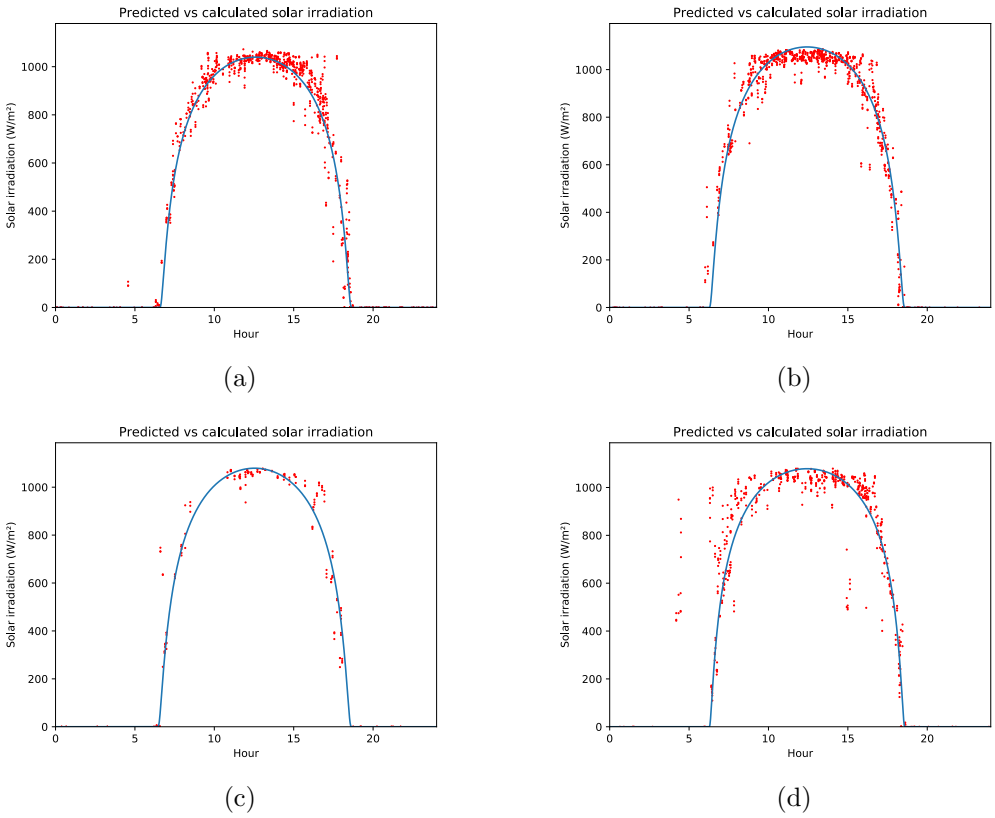


Figure 4.16: Comparing the calculation results against the prediction results for specific days. The blue graph represents the calculation, while each red dot is a test image with a specific predicted value. Four different days, spread over three years, have been chosen to give a fair representation of the results. (a) 13-05-2011. (b) 17-11-2011. (c) 24-09-2010. (d) 17-11-2012.

days score better than others (for the example in figure 4.16, the graph of 13-05-2011 appears to provide the cleanest results), but all graphs maintain good results to a certain degree. Heavy outliers still exist, however, and are worth discussing.

Figure 4.17 shows six of the heavier outliers in the test set, and table 4.7 gives the exact values for predicted and calculated solar irradiation, as well as the corresponding capture times. It appears that in this case, all six outliers occured in the late afternoon/early evening. Apparently, the model has more trouble predicting values in the late afternoon as opposed to values around noon. Possibly, the steep descent of solar irradiation in the afternoon (see also figure 4.16) is a cause for this issue. Although that may be a part of the issue, the position of the Sun[6] may also pose a problem. In figure 4.17a (outlier 1) for example, only a small part of the sky is visible as a wildebeest has inconveniently placed itself right in front of the camera. That visible part of the sky appears to be very bright (which the model might correlate with a high solar irradiation), while at 17:43'o clock the solar irradiation has already significantly decreased. Similarly, in figure 4.17b the camera captures what appears the sunset, again correlating the resulting brightness with a too high solar irradiation. Figure 4.17c appears to be a plain misprediction of the CNN, as the image is moderately bright corresponding to a moderate solar irradiation. In the meanwhile, the CNN calculated enough solar irradiation for noon. Outliers in the figures 4.17d and 4.17f suffer from the same problem. While the CNN appears to somewhat correctly predict the solar irradiation (the sky is overcast, obscuring much of the sunlight), the calculation model expected much higher values. These are two cases where an unclear sky causes the *clear-sky* model to fail, some of the noise expected from a mathematical model. Very likely, these two cases are not

[6]It is impossible to know the exact orientation of the camera (N,S,E,W), so it is possible that the the Sun is shining directly into the camera. There is no way to validate this, however.

Table 4.7: Calculated (using the clear-sky model) and predicted (using the regression CNN) values for solar irradiation as well as the time of image capture for six outliers.

	Clear-sky model (W/m^2)	Regression CNN (W/m^2)	Time (hh:mm:ss)
Outlier 1	522.59	1033.04	17:43:16
Outlier 2	234.18	661.92	18:17:40
Outlier 3	582.15	1014.37	17:37:55
Outlier 4	829.02	380.85	16:32:33
Outlier 5	630.79	191.32	17:29:23
Outlier 6	770.94	257.39	16:53:46

the only cases. Finally, figure 4.17e gives another case where the model simply fails to correctly predict the solar irradiation, without any clear cues as to why that is the case.

4.3.2 Exposure time

Where solar irradiation has a clear representation of its various values throughout the day, with for example a maximum at noon of around 1000 W/m^2. This intuition is missing for exposure time however, as throughout the day the exposure time can take many different values (which will be demonstrated later on in this section). In order to get some idea of the distribution of exposure time values, figure 4.18 gives a histogram plotting the distribution of the inverse exposure time (also called shutter speed) for the test set. The shutter speed simply is the result of 1 / exposure time. This inverting is necessary to train a CNN on the exposure times, as taking the value in seconds rather than $seconds^{-1}$ resulted in such small values that the network was unable to train well. It should be noted that the first bar in figure 4.18 contains mainly (if not only) inverted exposure times corresponding to the nighttime. Each image at nighttime, captured using a flash, has an inverse exposure time of 122 s^{-1} (or a $1/122 = 0.0082$ s exposure time). Therefore, since these are also the smallest inverse exposure time values, the first bar in the histogram contains all nighttime images with an inverse exposure time of $1/122$ s^{-1}. Consequently, it appears that the daytime images are usually captured with inverse exposure time values between 122 and 3000 s^{-1}, but larger values exist (with the highest inverse exposure time in the test set being 6766 s^{-1}).

Keeping the distribution of figure 4.18, it is now a possibility to discuss the results of the exposure time regression CNN. Figure 4.19 gives these for the test set, as the absolute error plotted in a histogram and a CDF. Again, table 4.8 gives a summary of the resulting CDF. From the histograms, it is clear that the vast majority of the absolute errors are values in the range [0,100]. When comparing table 4.6 and table 4.8, it appears that the difference in error CDF between the two parameters is initially very small, but that this value slowly increases as the CDF increases. While these tables of course belong to different parameters with different units, which implies that they cannot be simply compared, this gives some insight in the trends for the shutter speed. More specifically, it appears that the results of both parameters behave similarly. They have a large majority of absolute errors that are relatively small, as well as some errors that are significantly larger. For shutter speed, these errors can grow very large indeed (the largest absolute error is 5073.49 s^{-1}). As errors of this size are limited however, and as the majority of the errors is relatively small (the maximum values for shutter speed are much greater than those of solar irradiation), the results of the regression CNN appear to be satisfactory.

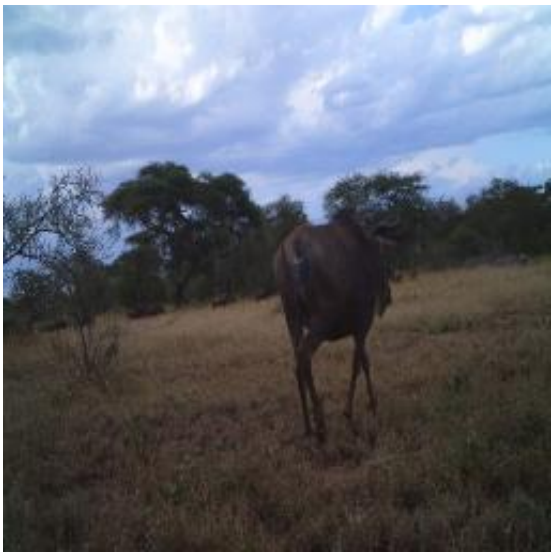
Table 4.8: Summary of the Cumulative Distribution Function of the absolute test errors using the CNN regression network for predicting shutter speed.

Percentages	10%	20%	30%	40%	50%	60%	70%	80%	90%
Error (s^{-1})	1.51	7.59	14.83	23.00	32.44	43.78	58.45	78.57	112.71

(a) Outlier 1

(b) Outlier 2

(c) Outlier 3



(d) Outlier 4

(e) Outlier 5

(f) Outlier 6

Figure 4.17: Six pictures for which the predicted and calculated solar irradiation differed. Table 4.7 gives more insight in the exact differences.

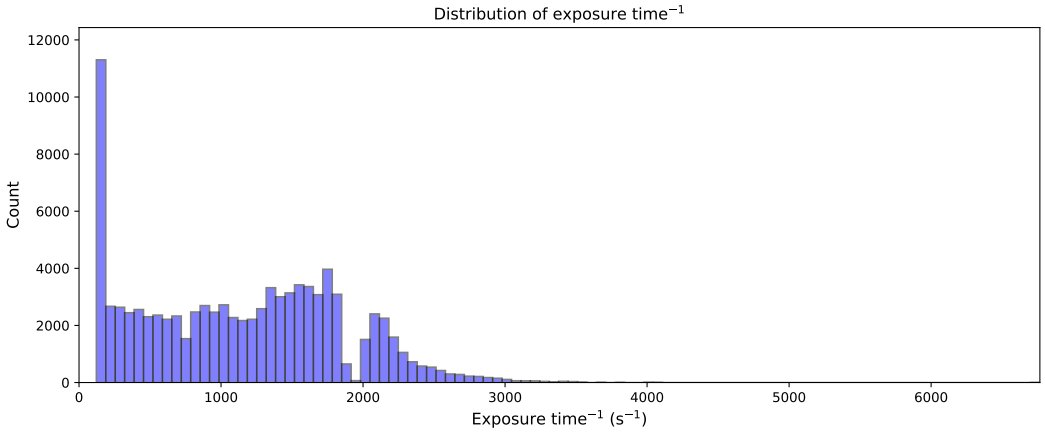
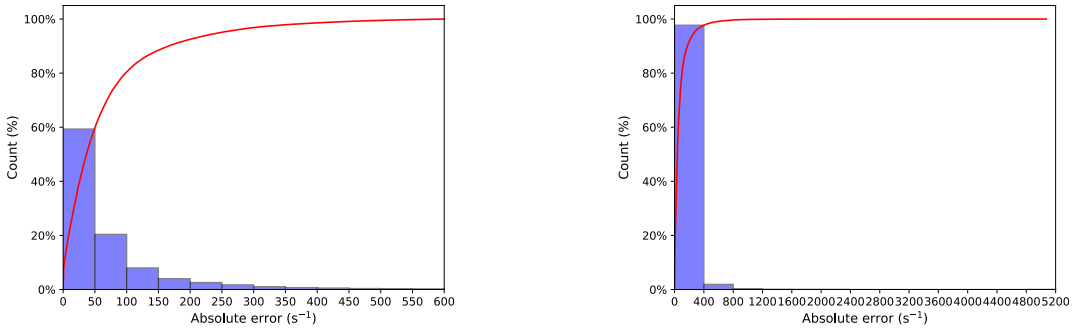


Figure 4.18: Distribution of the inverse exposure time or shutter speed for the test set. Note that while most values lie in the range of [122-3000], some much larger values exist as well.



(a) Absolute error up to 600 s^{-1}.

(b) Entire range of absolute errors.

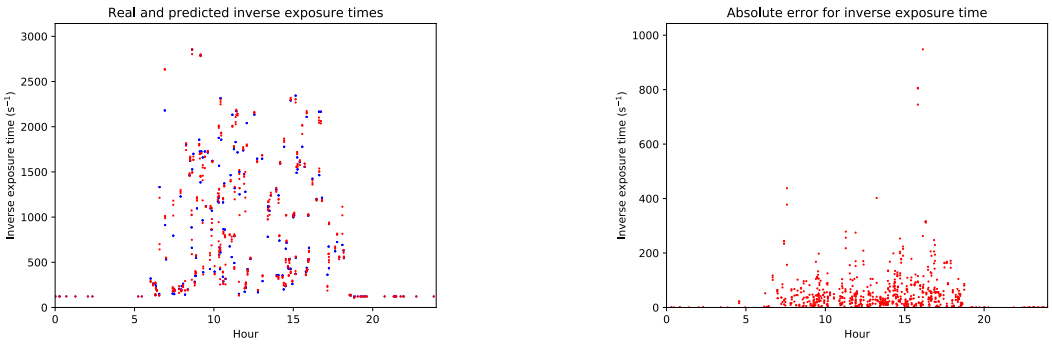
Figure 4.19: Histogram and CDF of the absolute error of the test results of the evaluation of the exposure time regression CNN. On the left a detailed view of the absolute errors up to 600 s^{-1}, while the left provides an overview of the entire error range. Note that the vast majority of the errors occurs in the first 50-100 s^{-1}, while very large errors occur as well.

To put these results into perspective, it would be useful to once again plot them for certain dates. However, since the real (and predicted) inverse exposure time values are much more erratic than the solar irradiation, it is hard to find a good overview plotting them. This issue is presented in figure 4.20a for 13-05-2011. Rather than examining the real and predicted values, it proves to be more useful to examine the absolute error values for a day (presented in figure 4.20b for the same day). Where the real and predicted values are very erratic[7], the absolute error values follow a certain trend. They are extremely small at nighttime (indicating that the 122 s^{-1} is learned very well), but increase at daytime (where actually different values for the inverse exposure time are plausible). While the absolute error is greater at daytime than at nighttime, it is still less that 50-100 s^{-1} in a large majority of the cases. As this day is a good representation of the the total dataset (and the errors on that dataset), there are a few outliers with large error values present as well. Considering that for this specific day 677 images[8] are used, these few outliers only concern about 1% of this day's data.

For completeness' sake, figure 4.21 gives the graphs corresponding to the other dates that were displayed for solar irradiation. The date in 2010 cannot be shown however, as the SG565

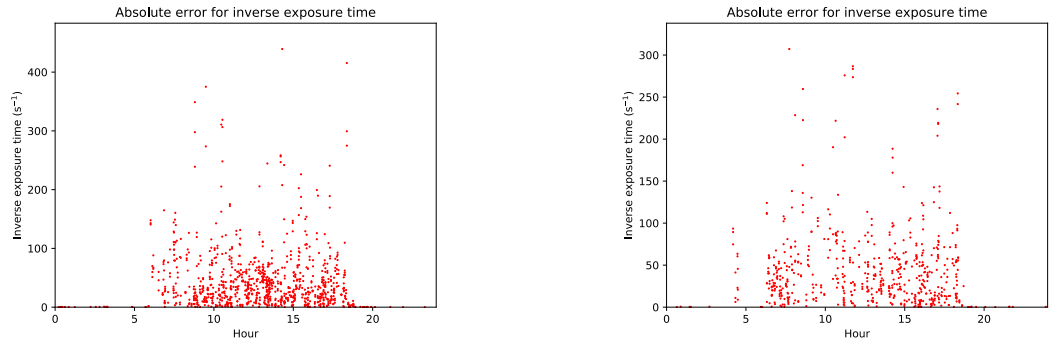
[7]This is because, unlike the clear-sky model, the actual solar irradiation at daytime can be erratic too. Clouds, shadows, position of the camera, ... all have an influence on the time of exposure while capturing images.

[8]The dataset sizes are smaller, as only images of the SG565 can be used.



(a) (b)

Figure 4.20: Left: Real (blue) and predicted (red) values for shutter speed for 13-05-2011. Right: Absolute error values for the shutter speed for 13-05-2011. Note that inverse exposure time at nighttime is 122 rather than zero, and that the predictions very closely approximate the real values.



(a) (b)

Figure 4.21: Left: Absolute errors for shutter speed at 17-11-2011. Right: Absolute errors for shutter speed at 17-11-2012.

cameras were not in use at that time. As can once again be clearly seen on these graphs, the majority of the errors happens at daytime and is smaller than 100 s^{-1}.

Finally, figure 4.22 gives some of the heavy outliers in the absolute error graph in figure 4.20b. These all appear to be members of the same capture event. Table 4.9 gives the predicted and real values for shutter speed, as well as the time of day. If the assumption is made that exposure time decreases as solar irradiation (or brightness) increases (to prevent saturation of the pixels in the camera), then shutter speed must increase with increasing brightness. In this case, it appears that the network recognizes the bright spot in the image and correlates it with a high shutter speed, while the actual exposure time is a lot smaller. This might be the case because the zebra in the image casts its shadow on the camera, reducing the solar irradiation incident on the camera.

Overall, it can be concluded that the predicted values for the exposure time (or inverse exposure time, which can be inverted back easily) are satisfactory and accurate enough to make actual predictions.

Table 4.9: Shutter speed prediction outliers.

Real	Predicted	Time (hh:mm:ss)
584	1390.54	15:49:33
584	1329.16	15:49:33
584	1388.19	15:49:33

(a) (b) (c)

Figure 4.22: Three images of the capture event that are very poorly predicted in the exposure time regression CNN at 13-5-2011.

4.4 Summary

In this chapter, experimental results have been presented. First, it has been proven that the calculated metadata (solar and lunar) are sufficiently accurate for the purposes of this research. Next, the classification of blanks and species is discussed. The trained networks are able to classify images with a very high accuracy and confidence, reaching state-of-the-art scores of over 90%. Additionally, some classified images have been provided to give more insight in the performance of the network and in its activations. Finally, the prediction of metadata using regression CNNs is discussed. It appears that MobileNetV2 is able to predict the solar irradiation or inverse exposure time from an image with small errors. Once again, practical examples give insight in the strengths and weaknesses of the regression networks. The next chapter will give methods to still improve the current results.

Chapter 5

Future Improvements

As in any research, there are a number of future improvements that could further increase performance. Since there are three major parts in this research, namely metadata calculation, classification and regression, each part will be discussed separately.

5.1 Metadata calculation

For the current calculation of the solar irradiation, the lunar phase, the lunar phase angle and the TOA lunar irradiation, three mathematical models were used. While each model is accurate up to a certain degree, it is hard to completely evaluate their accuracies as no observational data is available in two cases. Specifically, this is the case for the solar irradiation and the lunar irradiation, as the lunar phase and lunar age could be accurately validated. Moreover, it would be better still to simply use observational data instead of mathematical models. This would allow for a number of improvements. Firstly, there would be no more situations where the given solar irradiation would not correlate with the brightness of an image, as measured solar irradiation includes cloudiness and other meteorological events. The only factor observational data cannot always account for is the position of the sun as this would be dependent on the position of the sensor. Only if the sensor has the same field of view as the camera, the image brightness will completely correlate with the measured solar irradiation. Similarly, using measured values for the amount of moonlight/lunar irradiation would allow for even more accuracy. Additionally, this would take into account all wavelengths, rather than one selected wavelength λ_s.

5.2 Classification

As the classification task using CNNs is dependent on many different parameters, there are many different ways to improve it. Very likely, there still are a number of good improvements that have not been discovered yet. This section will give some adaptions to the current classification setup that may lead to improvements.
One of the main characteristics of this and other wildlife datasets is that there are many images that might be undesirable for training (see for example figure 3.2) due to their poor quality or their specific situation. Removing these form the training set (like using the blacklist) can help the network to better learn actual images. This follows the philosophy that, if the network can better recognize animals in baseline situations (without any factors making classification difficult), it should also be able to better recognize animals in difficult situations. It is for example hard to know that a visible trunk belongs to an elephant, if it is unknown that elephants have trunks. It is of course important that the network learns that difficult situations exist (take for example an image with an elephant's trunk visible throught the leaves of a bush), but it might perform better if its baseline training is better. One way to improve the training set would be to manually remove poor images, but that would be very labor intensive which would defeat the purpose of this research (improving efficiency when dealing with large wildlife image datasets). Another, potentially better way to improve the training set would be to use the currently trained

network to identify troublesome images. By then removing these images from the dataset and retraining the network, the accuracy could be improved.
A good dataset is an important requirement for machine learning, another is a strong learning architecture. While strong and commonly used networks architectures have been implemented in this research, newer and potentially better architectures exist too. Using one of these architectures (for example InceptionV3 [94], InceptopionV4 [95] or NASNet [54]) could help to further improve classification.
Unlike some other datasets, class imbalance is a significant issue for the Snapshot Serengeti dataset. While it has been somewhat examined, it has not been resolved in this project. Future improvements should strive to better classify very small classes such as Bushbuck or Caracal. Although image limits or class weights (as described in [58]) might slightly improve the classification of small classes, they do not improve the global performance of the network. Consequently, other methods should be explored. One such method could be to further divide the classification in different phases. Instead of simply classifying all species together, they could for example be classified in Zebra, Wildebeest, Thomsom's gazelle or other. This *other* class could then be further divided in the remaining species in one or multiple additional classification phases using networks that are trained on those species.
In the discussion of the inclusion of metadata in the network, a number of possibilities have been introduced that have the potential to increase the influence of metadata. One possibility is to try and introduce metadata in the first layers of a network, which would require careful consideration of the functioning of that network. On the other hand, it might prove useful to train and use a DNN after concatenating the metadata with the flattened output to help the network better understand the non-linear correlations between image data and metadata. Both methods should be examined.
Finally, one method that is sometimes used to slightly further increase classification is creating an *ensemble* by combining the classification outputs of many different networks (see for example [58]). As this would require many different architectures that need to be trained, which again requires a lot of time, this method has not been explored in this project.

5.3 Regression

Improving the efficiency of the regression CNNs could be done in several ways. Firstly, using observational data for solar irradiation instead of calculated values might improve the results. Besides that, first classifying between day and night and applying regression after that may narrow the range of values the regression CNN needs to handle. This may in turn lead to more accurate results. Finally, rather than limiting the experiments to regression, some metadata parameters (such as the camera model) could be predicted using a classification model. Experimenting with this metadata even more may result in more insight regarding the abilities of a CNN.

5.4 Summary

For each of the experiments improvements can be made. One of the most prominent possible improvements is the amelioration of the dataset by providing better samples with more information. The other discussed improvements are more specific to their experiment. In future research, the improvements can be examined.

Chapter 6

Conclusion

A lot of different properties have been researched in this project. It is therefore good to conclude by listing all the findings concerning these properties.

One major item that has been researched is the correlation between an image and specific metadata. Furthermore, it has been proven that some metadata can indeed be found in an image. Both solar irradiation as well as (inverse) exposure time can be predicted using a regression CNN. Moreover, these predictions are accurate as the large majority of the errors is very small (relative to the expected results). Including metadata as additional features in classification networks to improve accuracy does not seem to work, however. While simply concatenating these features to the output of the global average pooling layer does not change the results at all (indicating that it has no influence), using additional FC layers at the output of the network even proved to be detrimental to the performance.

Besides the influence of metadata, the use of different training strategies is examined. While brightness and contrast augmentation are common in the image recognition community, they appear to have no beneficial influence in training the Snapshot Serengeti dataset. Likewise, the use of CLAHE in preprocessing has been examined. Although CLAHE appears to help clarifying images for human viewers, it does not improve image classification at all. What makes a large difference, however, is the choice of architecture. MobileNetV2, being significantly smaller, not only always has lower accuracies than ResNet-50, but also has a harder to classify overly underrepresented classes. On the other hand, having only about a tenth of the parameters of ResNet-50, MobileNetV2 still obtains very high results. It therefore appears that MobileNetV2 is suitable for reliable image recognition applications on embedded systems.

Another important item is the comparison against related work. For the classification of blanks, the obtained results are slightly better than those obtained by [58]. When including the blacklisted images to the test set, the accuracies lower by about 1% and are just slightly lower than those of [58]. This indicates not only that the results are comparable to the state of art, it also indicates the influence of dubious or even wrongly labelled images in the dataset.
For the (blacklisted) species classification results, all obtained accuracies were again a little lower than those found in [58]. When observing the wrongly classified images, it is clear that many of the errors occur at images that would be difficult to classify for humans too. While the unbalanced dataset causes some species to be very hard to classify, the overall classification accuracy still is very high. Using an image limit to alleviate this unbalance only reduces the overall accuracy without significantly improving the classification of the underrepresented species.

Overall, it can be concluded that the obtained classification and regression accuracies are very high, and that the CNNs with the highest performance could be used for automation of labelling in biodiversity research.

Bibliography

[1] N. S. C. W. M. Kumar Duraiappah, A., *Ecosystems and Human Well-Being: Biodiversity Synthesis*. WRI, 2005, vol. 2005, https://www.biodiversitylibrary.org/bibliography/58170. [Online]. Available: https://www.biodiversitylibrary.org/item/119291

[2] F. Rovero, F. Zimmermann, D. Berzi, and P. Meek, ""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*, vol. 24, 08 2013.

[3] G. Chen, T. X. Han, Z. He, R. Kays, and T. Forrester, "Deep convolutional neural network based species recognition for wild animal monitoring," in *2014 IEEE International Conference on Image Processing (ICIP)*, Oct 2014, pp. 858–862.

[4] A. Swanson, M. Kosmala, C. Lintott, R. Simpson, A. Smith, and C. Packer, "Snapshot serengeti, high-frequency annotated camera trap images of 40 mammalian species in an african savanna," 2015. [Online]. Available: https://www.nature.com/articles/sdata201526

[5] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," *arXiv e-prints*, p. arXiv:1801.04381, Jan 2018.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[7] A. Swanson, M. Kosmala, C. Lintott, R. Simpson, A. Smith, and C. Packer, "Data from: Snapshot serengeti, high-frequency annotated camera trap images of 40 mammalian species in an african savanna," 2015. [Online]. Available: https://datadryad.org/resource/doi:10.5061/dryad.5pt92

[8] S. Africa, "Tanzania," 2018. [Online]. Available: http://songaafrica.com/tanzania

[9] B. world, "Serengeti national park," 2019. [Online]. Available: https://www.beautifulworld.com/africa/tanzania/serengeti-national-park/

[10] Zooniverse, "Snapshot serengeti," 2019. [Online]. Available: https://www.zooniverse.org/projects/zooniverse/snapshot-serengeti

[11] H. O. Products, *Digital Scouting Camera User's Manual SG565FV*, Norcross, GA, USA, 2019, http://www.hcooutdoors.com/media/wysiwyg/findme/SG565FV%20User's%20Manual.pdf.

[12] G. Ifrah, *The Universal History of Computing: From the Abacus to the Quantum Computer*. John Wiley & Sons, Inc, 2001.

[13] "Algorithm." [Online]. Available: https://en.wikipedia.org/wiki/Algorithm

[14] R. Cooke, *The History of Mathematics: A Brief Course*. Wiley, 2011.

[15] M. E. O'Neill, "The Genuine Sieve of Eratosthenes," *Journal of Functional Programming*, vol. 19, no. 1, pp. 95–106, 2009.

[16] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of Research and Development*, vol. 3, no. 3, pp. 210–229, July 1959.

[17] T. M. Mitchell, *Machine Learning*, 1st ed., ser. McGraw-Hill series in computer science. McGraw-Hill, 1997.

[18] E. Alpaydin, *Adaptive Computation and Machine Learning*, T. Dietterich, C. Bishop, D. Heckerman, M. Jordan, and M. Kearns, Eds. Cambrige: MIT Press, 2014.

[19] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85 – 117, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608014002135

[20] A. Ng, "Machine Learning." [Online]. Available: https://www.coursera.org/learn/machine-learning/

[21] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*. Berkeley, Calif.: University of California Press, 1967, pp. 281–297. [Online]. Available: https://projecteuclid.org/euclid.bsmsp/1200512992

[22] L. Fei Fei, A. Kartpathy, and J. Johnson, "Stanford University CS231n: Convolutional Neural Networks for Visual Recognition." [Online]. Available: http://cs231n.stanford.edu/2016/syllabus

[23] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural Networks: Tricks of the Trade, This Book is an Outgrowth of a 1996 NIPS Workshop*. London, UK, UK: Springer-Verlag, 1998, pp. 9–50.

[24] M. A. Nielsen, *Neural Networks and Deep Learning*. Determination Press, 2015.

[25] A. Vehbi Olgac and B. Karlik, "Performance analysis of various activation functions in generalized mlp architectures of neural networks," *International Journal of Artificial Intelligence And Expert Systems*, vol. 1, pp. 111–122, 02 2011.

[26] "What is an Activation Function?" [Online]. Available: https://deepai.org/machine-learning-glossary-and-terms/activation-function

[27] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *MCSS*, vol. 2, pp. 303–314, 1989.

[28] R. Lippmann, "An introduction to computing with neural nets," *IEEE ASSP Magazine*, vol. 4, no. 2, pp. 4–22, Apr 1987.

[29] H. Sak, A. W. Senior, and F. Beaufays, "Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition," *CoRR*, vol. abs/1402.1128, 2014. [Online]. Available: http://arxiv.org/abs/1402.1128

[30] A. Sherstinsky, "Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network," *CoRR*, vol. abs/1808.03314, 2018. [Online]. Available: http://arxiv.org/abs/1808.03314

[31] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," *CoRR*, vol. abs/1811.03378, 2018. [Online]. Available: http://arxiv.org/abs/1811.03378

[32] J. Han and C. Moraga, "The influence of the sigmoid function parameters on the speed of backpropagation learning," in *From Natural to Artificial Neural Computation*, J. Mira and F. Sandoval, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 195–201.

[33] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *CoRR*, vol. abs/1710.05941, 2017. [Online]. Available: http://arxiv.org/abs/1710.05941

[34] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ser. ICML'10. USA: Omnipress, 2010, pp. 807–814. [Online]. Available: http://dl.acm.org/citation.cfm?id=3104322.3104425

[35] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[36] S. Hochreiter, Y. Bengio, and P. Frasconi, "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," in *Field Guide to Dynamical Recurrent Networks*, J. Kolen and S. Kremer, Eds. IEEE Press, 2001.

[37] M. D. Zeiler, M. Ranzato, R. Monga, M. Z. Mao, K. Yang, Q. V. Le, P. Nguyen, A. W. Senior, V. Vanhoucke, J. Dean, and G. E. Hinton, "On rectified linear units for speech processing," *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 3517–3521, 2013.

[38] S. Linnainmaa, "Taylor expansion of the accumulated rounding error," *BIT Numerical Mathematics*, vol. 16, pp. 146 – 160, 1976.

[39] Y. LeCun, "A theoretical framework for back-propagation," in *Proceedings of the 1988 Connectionist Models Summer School*, D. Touretzky, G. Hinton, and T. Sejnowski, Eds. CMU, Pittsburgh, Pa: Morgan Kaufmann, 1988, pp. 21–28.

[40] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, Winter 1989.

[41] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, November 1998.

[42] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[43] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[44] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[45] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014. [Online]. Available: http://arxiv.org/abs/1409.1556

[46] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *CoRR*, vol. abs/1409.4842, 2014. [Online]. Available: http://arxiv.org/abs/1409.4842

[47] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Computer Vision and Pattern Recognition (CVPR)*, 2015. [Online]. Available: http://arxiv.org/abs/1409.4842

[48] "Kernel (image processing)," Jan 2019. [Online]. Available: https://en.wikipedia.org/wiki/Kernel_(image_processing)

[49] S. L. Eddins, R. C. Gonzalez, and R. E. Woods, *Digital image processing using MATLAB*. Upper Saddle River: Pearson Education, 2004.

[50] J. S. Bridle, *Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition*, 01 1990, pp. 227–236.

[51] M. Lin, Q. Chen, and S. Yan, "Network in network," *CoRR*, vol. abs/1312.4400, 2013. [Online]. Available: http://arxiv.org/abs/1312.4400

[52] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015. [Online]. Available: http://arxiv.org/abs/1502.03167

[53] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, vol. abs/1704.04861, 2017. [Online]. Available: http://arxiv.org/abs/1704.04861

[54] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," *CoRR*, vol. abs/1707.07012, 2017. [Online]. Available: http://arxiv.org/abs/1707.07012

[55] X. Ren, T. X. Han, and Z. He, "Ensemble video object cut in highly dynamic scenes," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, June 2013, pp. 1947–1954.

[56] F.-F. Li and P. Perona, "A bayesian hierarchical model for learning natural scene categories," in *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2 - Volume 02*, ser. CVPR '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 524–531. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2005.16

[57] A. G. Villa, A. Salazar, and F. Vargas, "Towards automatic wild animal monitoring: Identification of animal species in camera-trap images using very deep convolutional neural networks," *Ecological Informatics*, vol. 41, pp. 24 – 32, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1574954116302047

[58] M. S. Norouzzadeh, A. Nguyen, M. Kosmala, A. Swanson, M. S. Palmer, C. Packer, and J. Clune, "Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning," *Proceedings of the National Academy of Sciences*, 2018. [Online]. Available: http://www.pnas.org/content/early/2018/06/04/1719367115

[59] *2017 ASHRAE handbook: Fundamentals: SI edition*. ASHRAE, 2017.

[60] *Commission for Instruments and Methods of Observation: abridged final report of the eighth session, Mexico City, 19-30 October 1981*. Secretariat of the World Meteorological Organization, 1982.

[61] J. B. Tatum, *Chapter 6 The celestial sphere*. [Online]. Available: http://orca.phys.uvic.ca/~tatum/celmechs.html

[62] "Equatorial coordinate system," Jan 2019. [Online]. Available: https://en.wikipedia.org/wiki/Equatorial_coordinate_system

[63] S. Albarqouni and M. Hussein, "Enhanced model of one axis-two positions manual tracking photovoltaic panels for lighting projects in palestine," 01 2010.

[64] "Azimuth." [Online]. Available: https://en.wikipedia.org/wiki/Azimuth

[65] F. Kasten and A. Young, "Revised optical air-mass tables and approximation formula," *Applied Optics*, vol. 28, no. 22, pp. 4735–4738, 1989.

[66] P. Duffett-Smith, *Practical Astronomy with Your Calculator*, 4th ed. New York, NY, USA: Cambridge University Press, 2011.

[67] "Lunar phase," Jan 2019. [Online]. Available: https://en.wikipedia.org/wiki/Lunar_phase

[68] J. H. Meeus, *Astronomical Algorithms*. Willmann-Bell, Incorporated, 1991.

[69] "Phase angle (astronomy)," Jan 2019. [Online]. Available: https://en.wikipedia.org/wiki/Phase_angle_(astronomy)

[70] P. Seidelmann, U. S. N. O. N. A. Office, and G. B. N. A. Office, *Explanatory Supplement to the Astronomical Almanac*. University Science Books, 2006. [Online]. Available: https://books.google.be/books?id=uJ4JhGJANb4C

[71] R. H. Austin, B. Phillips, and D. Webb, "A method of calculating moonlight illuminance at the earth's surface," *Journal of Applied Ecology*, vol. 13, pp. 741–748, 12 1976.

[72] K. Krisciunas and B. E. Schaefer, "A model of the brightness of moonlight," *Publications of the Astronomical Society of the Pacific*, vol. 103, p. 1033, sep 1991. [Online]. Available: https://doi.org/10.1086%2F132921

[73] S. D. Miller and R. E. Turner, "A dynamic lunar spectral irradiance data set for npoess/viirs day/night band nighttime environmental applications," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 47, no. 7, pp. 2316–2329, July 2009.

[74] A. P. Lane and W. M. Irvine, "Monochromatic phase curves and albedos for the lunar disk," *aj*, vol. 78, p. 267, Apr. 1973.

[75] L. Krüger, *Konforme Abbildung des Erdellipsoids in der Ebene*, ser. Veröffentlichung des Königlich Preuszischen Geodätischen Instituts : Neue Folge. Leipzig: Teubner, 1912, vol. 52. [Online]. Available: urn:nbn:de:kobv:b103-krueger28

[76] C. F. F. Karney, "Transverse mercator with an accuracy of a few nanometers," *Journal of Geodesy*, vol. 85, no. 8, pp. 475–485, Aug 2011. [Online]. Available: https://doi.org/10.1007/s00190-011-0445-3

[77] T. Bieniek, "utm 0.4.2," April 2016. [Online]. Available: https://pypi.org/project/utm/

[78] "Ashrae climatic design conditions 2009/2013/2017," 2017. [Online]. Available: http://ashrae-meteo.info/

[79] "Serengeti," May 2007. [Online]. Available: https://en.wikipedia.org/wiki/Serengeti

[80] C. S. Cameras, "User's manual for dlc covert ii."

[81] "Exchangeable image file format for digital still cameras: Exif version 2.31 (english)," Camera & Imaging Products Association, Standard, July 2016.

[82] ianare, "Exifread 2.1.2," September 2015. [Online]. Available: https://pypi.org/project/ExifRead/

[83] F. Chollet, "Keras." [Online]. Available: https://keras.io/

[84] "TensorFlow." [Online]. Available: https://www.tensorflow.org/

[85] R. P. Singh and M. Dixit, "Histogram equalization: A strong technique for image enhancement," *International Journal of Signal Processing*, vol. 8, no. 8, pp. 345–352, 2015. [Online]. Available: https://pdfs.semanticscholar.org/2955/20205cfe5f0e1b3baba43f22458092fdf98e.pdf

[86] S. Pizer, E. Amburn, J. Austin, R. Cromartie, A. Geselowitz, T. Greer, B. Haar Romenij, ter, J. Zimmerman, and K. Zuiderveld, "Adaptive histogram equalization and its variations," *Computer Vision, Graphics, and Image Processing*, vol. 39, no. 3, pp. 355–368, 1987.

[87] S. Wiesler and H. Ney, "A convergence analysis of log-linear training," 12 2011, pp. 657–665.

[88] A. Kent, M. M. Berry, F. U. Luehrs Jr., and J. W. Perry, "Machine literature searching viii. operational criteria for designing information retrieval systems," *American Documentation*, vol. 6, no. 2, pp. 93–101, 1955. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/asi.5090060209

[89] Y. Sasaki, "The truth of the f-measure," *Teach Tutor Mater*, 01 2007.

[90] "Zonsopgang en zonsondergang hasselt 2011." [Online]. Available: https://www.sunrise-and-sunset.com/nl/sun/belgie/hasselt/2011/

[91] "Seasons in tanzania." [Online]. Available: https://seasonsyear.com/Tanzania

[92] E. Optics, "Imaging electronics 101: Understanding camera sensors for machine vision applications." [Online]. Available: https://www.edmundoptics.eu/resources/application-notes/imaging/understanding-camera-sensors-for-machine-vision-applications/

[93] B. Zhou, A. Khosla, L. A., A. Oliva, and A. Torralba, "Learning deep features for discriminative localization." *CVPR*, 2016. [Online]. Available: http://cnnlocalization.csail.mit.edu/Zhou_Learning_Deep_Features_CVPR_2016_paper.pdf

[94] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *CoRR*, vol. abs/1512.00567, 2015. [Online]. Available: http://arxiv.org/abs/1512.00567

[95] C. Szegedy, S. Ioffe, and V. Vanhoucke, "Inception-v4, inception-resnet and the impact of residual connections on learning," *CoRR*, vol. abs/1602.07261, 2016. [Online]. Available: http://arxiv.org/abs/1602.07261

Appendices

Appendix A

Classification results

This appendix contains all tables with complete species classification information regarding recall, precision and F1 score. Table A.1 gives an overview of all tables provided in this appendix. For more information regarding the setting, table 3.5 in section 3.3 can be examined.

Table A.1: Overview of classification result tables.

Table	Setting	Content
A.2	S1	MobileNetV2 with brightness and contrast augmentation
A.3	S2	ResNet-50 with brightness and contrast augmentation
A.4	S3	MobileNetV2 without brightness and contrast augmentation
A.5	S4	ResNet-50 without brightness and contrast augmentation
A.6	S5	MobileNetV2 with CLAHE augmentation
A.7	S6	MobileNetV2 with metadata features
A.8	S7	MobileNetV2 with metadata features with an additional FC layer
A.9	S8	MobileNetV2 with an additional FC layer
A.10	S9	ResNet-50 with metadata features
A.11	S10	MobileNetV2 with an image limit

Table A.2: MobileNetV2 - S1.

Class	Precision (%)	Recall (%)	F1 score (%)	Number of images
Aardvark	100.00	01.33	02.63	75
Aardwolf	00.00	00.00	00.00	46
Baboon	71.04	61.13	65.71	602
Bat-eared fox	34.48	10.31	15.87	97
Buffalo	86.22	80.37	83.20	4626
Bushbuck	00.00	00.00	00.00	42
Caracal	00.00	00.00	00.00	25
Cheetah	72.92	58.58	64.97	478
Civet	00.00	00.00	00.00	10
Dik-dik	52.04	62.47	56.78	429
Eland	80.96	63.60	71.24	956
Elephant	86.65	83.78	85.19	3563
Grant's gazelle	71.91	54.60	62.07	2916
Thomson's gazelle	88.09	94.61	91.24	15927
Genet	00.00	00.00	00.00	11
Giraffe	87.37	86.88	87.12	3057
Guineafowl	83.87	90.19	86.92	2957
Hare	29.75	33.96	31.72	106
Hartebeest	83.74	84.95	84.34	4433
Hippopotamus	79.27	85.29	82.17	435
Honey badger	00.00	00.00	00.00	13
Human	91.01	89.66	90.33	3434
Spotted hyena	70.63	68.53	69.56	1344
Striped hyena	00.00	00.00	00.00	34
Impala	77.18	84.83	80.82	3105
Jackal	28.57	02.58	04.73	155
Kori bustard	69.75	43.13	53.30	262
Leopard	00.00	00.00	00.00	52
Female lion	69.52	72.47	70.97	1108
Male lion	78.47	38.44	51.60	294
Mongoose	00.00	00.00	00.00	84
Ostrich	91.82	56.37	69.86	259
Other bird	61.46	53.48	57.20	1148
Porcupine	75.00	05.08	09.52	59
Reedbuck	67.91	69.93	68.91	572
Reptiles	100.00	81.82	90.00	33
Rhinoceros	00.00	00.00	00.00	9
Rodents	00.00	00.00	00.00	19
Secretary bird	61.84	53.11	57.14	177
Serval	62.50	07.63	13.61	131
Topi	79.51	58.21	67.22	840
Vervet monkey	74.24	40.83	52.69	120
Warthog	76.10	75.41	75.75	2761
Waterbuck	85.11	34.78	49.38	115
Wildcat	00.00	00.00	00.00	12
Wildebeest	94.38	96.45	95.40	33660
Zebra	92.03	94.77	93.38	21303
Zorilla	00.00	00.00	00.00	6
Weighted average	87.54	88.18	87.61	111900

Table A.3: ResNet-50 - S2.

Class	Precision (%)	Recall (%)	F1 score (%)	Number of images
Aardvark	62.35	70.67	66.25	75
Aardwolf	00.00	00.00	00.00	46
Baboon	81.33	73.09	76.99	602
Bat-eared fox	41.12	45.36	43.14	97
Buffalo	89.82	87.70	88.75	4626
Bushbuck	61.54	57.14	59.26	42
Caracal	100.00	08.00	14.81	25
Cheetah	74.49	75.73	75.10	478
Civet	00.00	00.00	00.00	10
Dik-dik	64.09	69.46	66.67	429
Eland	84.51	75.31	79.65	956
Elephant	89.73	90.51	90.12	3563
Grant's gazelle	72.60	60.25	65.85	2916
Thomson's gazelle	91.77	95.54	93.62	15927
Genet	00.00	00.00	00.00	11
Giraffe	90.87	91.17	91.02	3057
Guineafowl	89.84	92.70	91.25	2957
Hare	55.80	72.64	63.11	106
Hartebeest	87.88	87.80	87.84	4433
Hippopotamus	92.77	91.49	92.13	435
Honey badger	00.00	00.00	00.00	13
Human	92.97	93.54	93.25	3434
Spotted hyena	83.86	78.87	81.29	1344
Striped hyena	00.00	00.00	00.00	34
Impala	81.13	88.34	84.58	3105
Jackal	61.17	40.65	48.84	155
Kori bustard	65.33	56.11	60.37	262
Leopard	68.75	21.15	32.35	52
Female lion	80.30	83.48	81.86	1108
Male lion	73.54	55.78	63.44	294
Mongoose	38.30	21.43	27.48	84
Ostrich	89.95	72.59	80.34	259
Other bird	75.05	61.85	67.81	1148
Porcupine	75.86	74.58	75.21	59
Reedbuck	74.95	70.10	72.45	572
Reptiles	78.57	100.00	88.00	33
Rhinoceros	00.00	00.00	00.00	9
Rodents	00.00	00.00	00.00	19
Secretary bird	75.62	68.36	71.81	177
Serval	50.54	35.88	41.96	131
Topi	76.38	74.29	75.32	840
Vervet monkey	56.38	70.00	62.45	120
Warthog	84.53	81.13	82.79	2761
Waterbuck	83.95	59.13	69.39	115
Wildcat	00.00	00.00	00.00	12
Wildebeest	96.54	97.49	97.01	33660
Zebra	95.51	96.39	95.95	21303
Zorilla	00.00	00.00	00.00	6
Weighted average	91.15	91.46	91.23	111900

Table A.4: MobileNetV2 - S3.

Class	Precision (%)	Recall (%)	F1 score (%)	Number of images
Aardvark	40.00	02.67	05.00	75
Aardwolf	00.00	00.00	00.00	46
Baboon	72.89	67.44	70.06	602
Bat-eared fox	29.63	08.25	12.90	97
Buffalo	85.27	83.25	84.25	4626
Bushbuck	00.00	00.00	00.00	42
Caracal	00.00	00.00	00.00	25
Cheetah	72.05	66.32	69.06	478
Civet	00.00	00.00	00.00	10
Dik-dik	57.50	64.34	60.73	429
Eland	83.98	67.47	74.83	956
Elephant	86.55	85.21	85.87	3563
Grant's gazelle	68.81	55.32	61.33	2916
Thomson's gazelle	89.68	94.44	92.00	15927
Genet	00.00	00.00	00.00	11
Giraffe	89.36	87.37	88.36	3057
Guineafowl	85.36	90.70	87.95	2957
Hare	30.56	41.51	35.20	106
Hartebeest	85.60	85.72	85.66	4433
Hippopotamus	81.55	87.36	84.35	435
Honey badger	00.00	00.00	00.00	13
Human	89.65	89.78	89.71	3434
Spotted hyena	74.88	71.65	73.23	1344
Striped hyena	00.00	00.00	00.00	34
Impala	77.27	84.73	80.83	3105
Jackal	39.13	05.81	10.11	155
Kori bustard	65.05	46.18	54.02	262
Leopard	00.00	00.00	00.00	52
Female lion	74.95	74.01	74.48	1108
Male lion	79.76	45.58	58.01	294
Mongoose	00.00	00.00	00.00	84
Ostrich	92.90	60.62	73.36	259
Other bird	62.70	55.49	58.87	1148
Porcupine	91.67	18.64	30.99	59
Reedbuck	67.07	67.31	67.19	572
Reptiles	100.00	81.82	90.00	33
Rhinoceros	00.00	00.00	00.00	9
Rodents	00.00	00.00	00.00	19
Secretary bird	71.71	61.58	66.26	177
Serval	37.78	12.98	19.32	131
Topi	82.06	61.55	70.34	840
Vervet monkey	81.97	41.67	55.25	120
Warthog	78.45	77.11	77.77	2761
Waterbuck	82.76	41.74	55.49	115
Wildcat	00.00	00.00	00.00	12
Wildebeest	94.86	96.68	95.76	33660
Zebra	92.70	95.46	94.06	21303
Zorilla	00.00	00.00	00.00	6
Weighted average	88.26	88.92	88.43	111900

Table A.5: ResNet-50 - S4.

Class	Precision (%)	Recall (%)	F1 score (%)	Number of images
Aardvark	75.38	65.33	70.00	75
Aardwolf	16.67	02.17	03.85	46
Baboon	78.95	74.75	76.79	602
Bat-eared fox	44.34	48.45	46.31	97
Buffalo	88.70	88.09	88.39	4626
Bushbuck	75.00	57.14	64.86	42
Caracal	00.00	00.00	00.00	25
Cheetah	76.79	71.97	74.30	478
Civet	00.00	00.00	00.00	10
Dik-dik	66.52	70.40	68.40	429
Eland	82.37	75.73	78.91	956
Elephant	91.37	89.11	90.22	3563
Grant's gazelle	67.42	61.66	64.41	2916
Thomson's gazelle	92.31	94.96	93.62	15927
Genet	00.00	00.00	00.00	11
Giraffe	89.28	90.45	89.86	3057
Guineafowl	89.78	92.39	91.07	2957
Hare	57.81	69.81	63.25	106
Hartebeest	88.12	88.38	88.25	4433
Hippopotamus	90.47	89.43	89.94	435
Honey badger	00.00	00.00	00.00	13
Human	91.63	93.04	92.33	3434
Spotted hyena	80.27	79.61	79.94	1344
Striped hyena	00.00	00.00	00.00	34
Impala	81.41	85.35	83.33	3105
Jackal	52.74	49.68	51.16	155
Kori bustard	64.03	61.83	62.91	262
Leopard	55.56	28.85	37.97	52
Female lion	79.84	80.42	80.13	1108
Male lion	73.28	57.82	64.64	294
Mongoose	42.59	27.38	33.33	84
Ostrich	77.42	74.13	75.74	259
Other bird	73.94	57.84	64.91	1148
Porcupine	75.00	66.10	70.27	59
Reedbuck	75.64	72.73	74.15	572
Reptiles	72.73	96.97	83.12	33
Rhinoceros	00.00	00.00	00.00	9
Rodents	00.00	00.00	00.00	19
Secretary bird	75.66	64.97	69.91	177
Serval	54.46	46.56	50.21	131
Topi	77.72	72.26	74.89	840
Vervet monkey	63.11	64.17	63.64	120
Warthog	84.28	82.14	83.20	2761
Waterbuck	82.76	62.61	71.29	115
Wildcat	00.00	00.00	00.00	12
Wildebeest	96.65	97.31	96.98	33660
Zebra	95.39	96.47	95.93	21303
Zorilla	00.00	00.00	00.00	6
Weighted average	90.93	91.24	91.05	111900

Table A.6: MobileNetV2 - S5.

Class	Precision (%)	Recall (%)	F1 score (%)	Number of images
Aardvark	50.00	04.00	07.41	75
Aardwolf	00.00	00.00	00.00	46
Baboon	69.32	66.45	67.85	602
Bat-eared fox	41.38	12.37	19.05	97
Buffalo	84.35	80.76	82.52	4626
Bushbuck	00.00	00.00	00.00	42
Caracal	00.00	00.00	00.00	25
Cheetah	75.41	67.36	71.16	478
Civet	00.00	00.00	00.00	10
Dik-dik	55.05	67.37	60.59	429
Eland	80.96	63.60	71.24	956
Elephant	86.24	84.09	85.15	3563
Grant's gazelle	71.25	53.64	61.20	2916
Thomson's gazelle	88.78	94.52	91.56	15927
Genet	00.00	00.00	00.00	11
Giraffe	88.27	86.62	87.44	3057
Guineafowl	86.48	89.55	87.99	2957
Hare	31.62	34.91	33.18	106
Hartebeest	84.24	85.02	84.63	4433
Hippopotamus	80.09	84.14	82.06	435
Honey badger	00.00	00.00	00.00	13
Human	89.44	90.24	89.84	3434
Spotted hyena	71.77	70.16	70.96	1344
Striped hyena	00.00	00.00	00.00	34
Impala	77.92	83.32	80.53	3105
Jackal	31.82	04.52	07.91	155
Kori bustard	57.35	44.66	50.21	262
Leopard	00.00	00.00	00.00	52
Female lion	72.68	70.13	71.38	1108
Male lion	72.67	37.07	49.10	294
Mongoose	00.00	00.00	00.00	84
Ostrich	88.41	55.98	68.56	259
Other bird	63.98	54.01	58.57	1148
Porcupine	100.00	10.17	18.46	59
Reedbuck	66.49	67.31	66.90	572
Reptiles	100.00	90.91	95.24	33
Rhinoceros	00.00	00.00	00.00	9
Rodents	00.00	00.00	00.00	19
Secretary bird	62.75	54.24	58.18	177
Serval	43.59	12.98	20.00	131
Topi	76.95	60.00	67.42	840
Vervet monkey	81.03	39.17	52.81	120
Warthog	79.28	74.83	76.99	2761
Waterbuck	77.94	46.09	57.92	115
Wildcat	00.00	00.00	00.00	12
Wildebeest	93.92	96.48	95.18	33660
Zebra	92.08	94.98	93.50	21303
Zorilla	00.00	00.00	00.00	6
Weighted average	87.54	88.26	87.71	111900

Table A.7: MobileNetV2 - S6.

Class	Precision (%)	Recall (%)	F1 score (%)	Number of images
Aardvark	20.00	01.33	02.50	75
Aardwolf	00.00	00.00	00.00	46
Baboon	71.23	66.61	68.84	602
Bat-eared fox	25.00	05.15	08.55	97
Buffalo	86.48	81.58	83.96	4626
Bushbuck	100.00	02.38	04.65	42
Caracal	00.00	00.00	00.00	25
Cheetah	74.42	67.57	70.83	478
Civet	00.00	00.00	00.00	10
Dik-dik	57.91	65.73	61.57	429
Eland	80.51	65.69	72.35	956
Elephant	87.42	85.24	86.32	3563
Grant's gazelle	74.68	54.73	63.17	2916
Thomson's gazelle	90.18	94.88	92.47	15927
Genet	00.00	00.00	00.00	11
Giraffe	88.73	87.60	88.16	3057
Guineafowl	86.97	90.02	88.47	2957
Hare	26.15	32.08	28.81	106
Hartebeest	84.35	86.17	85.25	4433
Hippopotamus	78.39	85.06	81.59	435
Honey badger	00.00	00.00	00.00	13
Human	91.15	89.11	90.12	3434
Spotted hyena	75.39	72.02	73.67	1344
Striped hyena	00.00	00.00	00.00	34
Impala	78.00	85.31	81.50	3105
Jackal	55.00	07.10	12.57	155
Kori bustard	60.42	44.27	51.10	262
Leopard	00.00	00.00	00.00	52
Female lion	72.70	74.73	73.70	1108
Male lion	74.68	40.14	52.21	294
Mongoose	00.00	00.00	00.00	84
Ostrich	92.86	60.23	73.07	259
Other bird	64.80	50.52	56.78	1148
Porcupine	92.31	20.34	33.33	59
Reedbuck	65.07	69.06	67.01	572
Reptiles	90.91	90.91	90.91	33
Rhinoceros	00.00	00.00	00.00	9
Rodents	00.00	00.00	00.00	19
Secretary bird	62.82	55.37	58.86	177
Serval	42.86	16.03	23.33	131
Topi	78.68	62.38	69.59	840
Vervet monkey	75.47	33.33	46.24	120
Warthog	77.04	77.80	77.42	2761
Waterbuck	79.10	46.09	58.24	115
Wildcat	00.00	00.00	00.00	12
Wildebeest	94.07	97.16	95.59	33660
Zebra	92.73	95.15	93.92	21303
Zorilla	00.00	00.00	00.00	6
Weighted average	88.26	88.92	88.38	111900

Table A.8: MobileNetV2 - S7.

Class	Precision (%)	Recall (%)	F1 score (%)	Number of images
Aardvark	00.00	00.00	00.00	75
Aardwolf	00.00	00.00	00.00	46
Baboon	65.65	67.61	66.61	602
Bat-eared fox	00.00	00.00	00.00	97
Buffalo	83.83	82.71	83.26	4626
Bushbuck	00.00	00.00	00.00	42
Caracal	00.00	00.00	00.00	25
Cheetah	65.12	67.57	66.32	478
Civet	00.00	00.00	00.00	10
Dik-dik	38.59	62.70	47.78	429
Eland	80.91	63.39	71.09	956
Elephant	87.12	83.13	85.08	3563
Grant's gazelle	74.03	54.05	62.48	2916
Thomson's gazelle	88.86	94.90	91.78	15927
Genet	00.00	00.00	00.00	11
Giraffe	90.10	86.03	88.02	3057
Guineafowl	85.63	90.06	87.79	2957
Hare	00.00	00.00	00.00	106
Hartebeest	85.93	85.38	85.65	4433
Hippopotamus	79.96	84.37	82.10	435
Honey badger	00.00	00.00	00.00	13
Human	90.01	90.51	90.26	3434
Spotted hyena	70.55	70.24	70.40	1344
Striped hyena	00.00	00.00	00.00	34
Impala	76.58	85.83	80.94	3105
Jackal	00.00	00.00	00.00	155
Kori bustard	42.01	35.11	38.25	262
Leopard	00.00	00.00	00.00	52
Female lion	72.41	75.81	74.07	1108
Male lion	76.60	36.73	49.66	294
Mongoose	00.00	00.00	00.00	84
Ostrich	86.39	56.37	68.22	259
Other bird	61.11	53.92	57.29	1148
Porcupine	00.00	00.00	00.00	59
Reedbuck	62.54	70.63	66.34	572
Reptiles	100.00	21.21	35.00	33
Rhinoceros	00.00	00.00	00.00	9
Rodents	00.00	00.00	00.00	19
Secretary bird	64.71	18.64	28.95	177
Serval	00.00	00.00	00.00	131
Topi	77.76	61.19	68.49	840
Vervet monkey	00.00	00.00	00.00	120
Warthog	76.99	75.73	76.36	2761
Waterbuck	100.00	00.87	01.72	115
Wildcat	00.00	00.00	00.00	12
Wildebeest	94.36	96.41	95.38	33660
Zebra	92.33	94.90	93.60	21303
Zorilla	00.00	00.00	00.00	6
Weighted average	87.30	88.26	87.57	111900

Table A.9: MobileNetV2 - S8.

Class	Precision (%)	Recall (%)	F1 score (%)	Number of images
Aardvark	00.00	00.00	00.00	75
Aardwolf	00.00	00.00	00.00	46
Baboon	69.98	65.45	67.64	602
Bat-eared fox	00.00	00.00	00.00	97
Buffalo	84.54	81.30	82.89	4626
Bushbuck	00.00	00.00	00.00	42
Caracal	00.00	00.00	00.00	25
Cheetah	71.46	63.39	67.18	478
Civet	00.00	00.00	00.00	10
Dik-dik	40.03	66.43	49.96	429
Eland	79.17	64.02	70.79	956
Elephant	87.12	83.89	85.47	3563
Grant's gazelle	72.42	55.28	62.70	2916
Thomson's gazelle	89.56	94.05	91.75	15927
Genet	00.00	00.00	00.00	11
Giraffe	88.92	86.88	87.89	3057
Guineafowl	83.58	90.56	86.93	2957
Hare	33.33	01.89	03.57	106
Hartebeest	84.72	85.54	85.13	4433
Hippopotamus	74.35	85.98	79.74	435
Honey badger	00.00	00.00	00.00	13
Human	90.21	89.90	90.05	3434
Spotted hyena	64.88	71.35	67.97	1344
Striped hyena	00.00	00.00	00.00	34
Impala	77.41	85.19	81.11	3105
Jackal	00.00	00.00	00.00	155
Kori bustard	37.35	35.50	36.40	262
Leopard	00.00	00.00	00.00	52
Female lion	66.32	74.82	70.31	1108
Male lion	78.86	32.99	46.52	294
Mongoose	00.00	00.00	00.00	84
Ostrich	89.53	59.46	71.46	259
Other bird	58.22	50.00	53.80	1148
Porcupine	00.00	00.00	00.00	59
Reedbuck	61.15	67.13	64.00	572
Reptiles	00.00	00.00	00.00	33
Rhinoceros	00.00	00.00	00.00	9
Rodents	00.00	00.00	00.00	19
Secretary bird	50.00	02.26	04.32	177
Serval	00.00	00.00	00.00	131
Topi	77.45	65.00	70.68	840
Vervet monkey	00.00	00.00	00.00	120
Warthog	78.07	75.15	76.58	2761
Waterbuck	00.00	00.00	00.00	115
Wildcat	00.00	00.00	00.00	12
Wildebeest	94.46	96.62	95.53	33660
Zebra	92.37	94.93	93.63	21303
Zorilla	00.00	00.00	00.00	6
Weighted average	87.08	88.14	87.44	111900

Table A.10: ResNet-50 - S9.

Class	Precision (%)	Recall (%)	F1 score (%)	Number of images
Aardvark	75.00	60.00	66.67	75
Aardwolf	00.00	00.00	00.00	46
Baboon	83.65	73.92	78.48	602
Bat-eared fox	42.22	39.18	40.64	97
Buffalo	87.87	88.15	88.01	4626
Bushbuck	70.00	50.00	58.33	42
Caracal	00.00	00.00	00.00	25
Cheetah	78.80	71.55	75.00	478
Civet	00.00	00.00	00.00	10
Dik-dik	62.66	68.07	65.25	429
Eland	84.66	75.63	79.89	956
Elephant	90.45	88.52	89.48	3563
Grant's gazelle	69.80	60.63	64.89	2916
Thomson's gazelle	92.07	95.35	93.68	15927
Genet	00.00	00.00	00.00	11
Giraffe	90.88	91.27	91.07	3057
Guineafowl	91.66	92.49	92.07	2957
Hare	58.33	72.64	64.71	106
Hartebeest	87.08	87.86	87.47	4433
Hippopotamus	90.18	90.80	90.49	435
Honey badger	00.00	00.00	00.00	13
Human	92.74	92.57	92.66	3434
Spotted hyena	81.85	78.87	80.33	1344
Striped hyena	00.00	00.00	00.00	34
Impala	81.00	85.41	83.15	3105
Jackal	48.00	46.45	47.21	155
Kori bustard	67.14	53.82	59.75	262
Leopard	73.68	26.92	39.44	52
Female lion	76.37	80.51	78.38	1108
Male lion	73.42	59.18	65.54	294
Mongoose	39.44	33.33	36.13	84
Ostrich	88.73	72.97	80.08	259
Other bird	71.24	59.76	64.99	1148
Porcupine	74.58	74.58	74.58	59
Reedbuck	69.62	72.90	71.22	572
Reptiles	97.06	100.00	98.51	33
Rhinoceros	00.00	00.00	00.00	9
Rodents	00.00	00.00	00.00	19
Secretary bird	76.77	67.23	71.69	177
Serval	53.06	39.69	45.41	131
Topi	80.97	71.43	75.90	840
Vervet monkey	66.02	56.67	60.99	120
Warthog	83.28	81.89	82.58	2761
Waterbuck	85.71	67.83	75.73	115
Wildcat	00.00	00.00	00.00	12
Wildebeest	96.47	97.54	97.00	33660
Zebra	95.49	96.48	95.98	21303
Zorilla	00.00	00.00	00.00	6
Weighted average	90.94	91.27	91.06	111900

Table A.11: MobileNetV2 - S10.

Class	Precision (%)	Recall (%)	F1 score (%)	Number of images
Aardvark	20.00	01.33	02.50	75
Aardwolf	00.00	00.00	00.00	46
Baboon	63.17	62.96	63.06	602
Bat-eared fox	40.74	22.68	29.14	97
Buffalo	71.84	81.06	76.17	4626
Bushbuck	00.00	00.00	00.00	42
Caracal	00.00	00.00	00.00	25
Cheetah	70.77	63.81	67.11	478
Civet	00.00	00.00	00.00	10
Dik-dik	50.64	64.10	56.58	429
Eland	72.54	61.61	66.63	956
Elephant	76.86	83.33	79.96	3563
Grant's gazelle	60.39	57.61	58.97	2916
Thomson's gazelle	89.60	90.95	90.27	15927
Genet	00.00	00.00	00.00	11
Giraffe	80.51	86.10	83.21	3057
Guineafowl	80.50	89.89	84.93	2957
Hare	29.01	44.34	35.07	106
Hartebeest	77.98	85.72	81.67	4433
Hippopotamus	80.40	82.99	81.67	435
Honey badger	00.00	00.00	00.00	13
Human	84.57	87.33	85.93	3434
Spotted hyena	61.30	70.01	65.37	1344
Striped hyena	00.00	00.00	00.00	34
Impala	73.24	84.86	78.62	3105
Jackal	38.10	10.32	16.24	155
Kori bustard	56.04	38.93	45.95	262
Leopard	100.00	01.92	03.77	52
Female lion	67.08	73.01	69.92	1108
Male lion	68.21	35.03	46.29	294
Mongoose	00.00	00.00	00.00	84
Ostrich	81.97	57.92	67.87	259
Other bird	53.54	51.39	52.44	1148
Porcupine	80.00	27.12	40.51	59
Reedbuck	62.44	64.51	63.46	572
Reptiles	100.00	90.91	95.24	33
Rhinoceros	00.00	00.00	00.00	9
Rodents	00.00	00.00	00.00	19
Secretary bird	62.24	50.28	55.62	177
Serval	46.15	13.74	21.18	131
Topi	72.34	58.21	64.51	840
Vervet monkey	79.63	35.83	49.43	120
Warthog	67.04	74.68	70.65	2761
Waterbuck	86.05	32.17	46.84	115
Wildcat	00.00	00.00	00.00	12
Wildebeest	95.30	91.07	93.14	33660
Zebra	92.47	91.08	91.77	21303
Zorilla	00.00	00.00	00.00	6
Weighted average	85.37	85.35	85.17	111900