Master in Computational Colour and Spectral Imaging (COSI)

Metadata Augmented Deep Neural Networks for Wild Animal Classification

Master Thesis Report

Presented by

Aslak Tøn

and defended at the

Norwegian University of Science and Technology

September 2023

Academic Supervisor(s): Ali Shariq Imran, Mohib Ullah
Jury Committee:

1. Damien Muselet, University of Jean Monnet, France
2. Dr. Sajib Saha, Commonwealth Scientific and Industrial Research Organisation, Australia

Submission of the thesis: 10th August 2023
Day of the oral defense: 5th September 2023

Abstract

Camera trap imaging has emerged as a valuable tool for modern wildlife surveillance, enabling researchers to monitor and study wild animals and their behaviors. However, a significant challenge in camera trap data analysis is the labor-intensive task of species classification from the captured images. Utilizing deep learning has been proven to be an effective solution to reduce the workload of ecological researchers.

This thesis proposes the usage of specific metadata, including temperature, location, and time, to enhance the established field of image classification. We demonstrate the effectiveness of this approach on a dataset centered on the Norwegian climate. Our models, compared against existing ones widely used in the field, demonstrated an increase in accuracy from 98.4% to 98.9%. While this increase may seem marginal, given that the models are already approaching perfect accuracy, we argue this improvement is significant.

Furthermore, we demonstrate the potential for improving models with metadata without the need for extra work in the data collection phase. Using deep learning models for scene recognition, we achieved high prediction accuracy in an ablative study focused on classification purely from the metadata in our dataset. This automated pipeline can be used in future comprehensive networks that incorporate both image data and metadata, which could significantly enhance the image classification of wild animals.

Keywords: *Wild animal detection, Wild animal classification, deep learning, data fusion.*

Acknowledgment

I would like to thank my supervisors, Ali Shariq Imran and Mohib Ullah, for guiding me through this masters thesis. Their willingness to discuss with me, and dedicate their time to my research has helped me gain better insight into the field, and utimately, create a significantly better work. Without their advice at key moments, this thesis would not have come to the same conclusions. Their knowledge also extended to the report, providing feedback to improve how I conveyed my contributions to the reader. This thesis would be significantly less friendly to the reader without the contributions of my supervisors.

Thank you to the Norwegian University of Science and Technology for providing me with tools to more efficiently perform my research. Tools were always provided with minimal delay, making for an enjoyable research experience.

I would also like to thank NINA and especially John Linell for allowing us to work with their dataset. This thesis would not have been possible without the data they willingly provided us.

Glossary

Table 1: *Glossary Table*

Term	Definition
Accuracy	The ratio of correct predictions to the total number of predictions made.
Bounding box	A rectangle drawn around the target object in an image used in object detection tasks.
Camera Trap	A stationary camera often fastened to a tree, used to capture animals in their natural habitat. Utilizes RGB sensors for daytime and IR sensors for nighttime capture.
Data Augmentation	Techniques used to increase the amount of data by adding slightly modified copies of already existing data. Discussed further in Section 2.5.3.
Data Fusion	The process of integrating data of different modalities. Discussed in Section 2.4.
Datetime	A 67-dimensional vector representing the month, day, and hour of a specific datapoint. See Section 4.1.5 for further information.
Deep learning	A branch of machine learning using multi-layered neural networks to interpret complex data patterns. See Section 2.2 for details.
False negative (FN)	An outcome where the model incorrectly predicts the negative class.
False negative rate (FNR)	The ratio of false negatives to the sum of false negatives and true positives.
False positive (FP)	An outcome where the model incorrectly predicts the positive class.
False positive rate (FPR)	The ratio of false positives to the sum of false positives and true negatives.
Label	In supervised learning, the 'answer' or 'result' portion of an example in the dataset.
One hot encoding	A representation of categorical data as binary vectors where each category is assigned a unique binary value.

Term	Definition
Precision	The ratio of true positives to the sum of true positives and false positives.
Recall	The ratio of true positives to the sum of true positives and false negatives.
Scene attribute	Presence or absence of a specific attribute to a scene. discussed in Section 4.1.5.
Scene descriptor	The actual scene the image is captured in. Also discussed in Section 4.1.5.
True negative (TN)	An outcome where the model correctly predicts the negative class.
True positive (TP)	An outcome where the model correctly predicts the positive class.
Viltkamera	(en) Wildlife camera, the name used for the Norwegian Camera Trap Project.
F_1 score	Harmonic mean of precision and recall, used to measure a model's performance.

Acronyms

Table 2: *Acronym Table*

Acronym	Definition
ADAM	ADAptive Moment estimation
AI	Artificial Intelligence
CBAM	Channel Block Attention Module
CNN	convolutional Neural Network
DL	Deep Learning
DNN	Deep Neural Network
GAN	Generative Adversarial Network
GMU	Gated Multimodal Unit
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
IoU	Intersection over Union, aka Jaccard Index
LILA BC	Labeled information Library of Alexandria: Biology and Conservation
LION	EvoLved Sign Momentum
LLM	Large Language Models
MLP	Multi-layer perceptron
MCBAM	Modified Channel Block Attention Module
NINA	Norwegian Institute for Nature Research (NO: Norsk institutt for naturforskning)
ReLU	Rectified Linear Unit
RGB	Red Green Blue – usually referring to images/camera handling visual spectrum
SGD	Stochastic Gradient Descent
SMOTE	Synthetic Minority Over-sampling Technique
SOTA	State Of The Art
SS	Snapshot Serengeti
UMAP	Uniform Manifold Approximation and Projection
WAC	Wild Animal Classification

Contents

CONTENTS

1 Introduction

1.1 Motivation

The human species has come to dominate the ecology of our planet over the last 300-400 years. This human expansion and utilization of natural resources has caused immense strain on the wild animals around the globe. Many factors play a role in the decrease in wildlife biodiversity, from human caused (Masson-Delmotte et al. (2021)) climate change (Pörtner et al. (2022)), deforestation (Lata et al. (2018)), and trafficked roads (Dean et al. (2019)) all play part in the reduction of wildlife populations, leading to a modern mass extinction event (Pievani (2014)). This rapid transformation of the natural habitats of wild animals affects the wild animals both in behavior and population Njamasi et al. (2022). The impact of humans on the planet is so widespread, a new geologic era has been created. This geologic era, dubbed "the anthropocene", is marked as a world dominated by human actions on the planet Lewis and Maslin (2015). The outsized impact humanity has on our ecosystems necessitates monitoring of wild animal habitats. This data gives researchers invaluable evidence to protect and manage decisions in order to maintain a diverse, sustainable & balanced ecosystem in the face of anthropogenic activities Berger et al. (2006), Patterson et al. (2008). For instance, due to good access to wildlife data, Australian researchers managed to chart out an estimated impact on the wildlife due to the 2019-20 wildfires in Australia (Hyman et al. (2020)). This information gave policymakers an opportunity to react more appropriately to preserve the wildlife in the region.

To monitor wildlife, one may employ several techniques. These vary from radio tracking (Habib et al. (2014)), to wireless sensor network tracking (Garcia-Sanchez et al. (2010)), remote sensing and Global Positioning Systems (GPS) (Recio et al. (2011)), radar methods (Flock and Green (1974)), and motion sensitive camera traps (Cordier et al. (2022)). This thesis will focus on the use of motion sensitive camera traps, or Wildlife Camera Traps (WCT). Camera traps is an efficient way of monitoring statistics on animals without requiring tagging animals, or disrupting the normal habitat significantly. A network of camera traps is a cheap and fast to

deploy monitoring method that can collect vast amounts of data about all animals inhabiting or migrating through the network region. This data is quite helpful for monitoring wild animal population and biodiversity. If the samples are taken over several seasons, it is also possible to infer the change in these variables. In addition, due to the pictographic nature of Camera Traps, it is also possible to track individuals of a given species. This subject level information can give us an even more detailed image of how the current status of a wild animal population. One of the major issues with camera trap project is the difficulty in extracting the relevant data from the images. Images needs to be tagged, labeled and sorted before real data analysis can be performed. Citizen science has been shown to be of great help here, achieving an accuracy of 96.6% when classifying for the Snapshot Serengeti project (Swanson et al. (2015)).

With the number of labeled samples available for Camera Trap projects, image classification using machine learning (ML) is becoming more realistic by the day. The field is not without challenges. Wild animals are not known for their photogenic postures, and lighting is rarely in an optimal spot, or even there in the case of nocturnal animals. However, we believe machine learning can prove to be great tool when properly tuned for Wild Animal Classification (WAC). Reducing the workload on scientists significantly, and letting them contribute by analyzing the data available via automatic WAC instead of spending significant research time labeling images.

This thesis focuses on a dataset produced by the Norwegian Institute for Nature Research (NINA), addressing the need for region-specific models for accurate wildlife detection and classification. Specifically, it will explore the potential benefits of including metadata in the classification problem using deep learning, highlighting the necessity of tailoring models to account for climatic and ecological variations.

The field of automatic WAC has already been explored on larger datasets like Snapshot Serengeti. However, the accuracy of the trained models can significantly decrease when transferred to different climatic regions, such as Norway, due to the stark contrast in biodiversity and landscape. In particular, the deep learning models trained on the Snapshot Serengeti (SS) dataset may not perform well in the Norwegian climate due to differences in animal species, their behaviors, and unique visual aspects in their respective environments.

1.2 Research Questions

The research questions, and sub-questions, we wish to explore can be summarized as follows;

1. How can metadata be effectively incorporated into deep learning models for

WAC?

   (a) Which types metadata provides the greatest impact on WAC?

   (b) How does the inclusion of metadata impact the performance of deep learning models in classifying images from wildlife camera traps (WCT)?

2. How does the performance of modified deep learning models compare to traditional architectures when applied to WCT image classification?

   (a) What classes can be sensibly grouped, using information from metadata, to increase model performance?

   (b) What modifications needs to be done on existing deep learning models to maximize performance for WAC in a Nordic setting?

1.3 Our Contributions

These are the key contributions of this thesis:

1. Our research involved the collection and preparation of 170 thousand image and metadata samples from the NINA Viltkamera dataset. We demonstrated how to effectively tackle issues with missing metadata by filling in the gaps wherever possible, and otherwise give a network context for when data was not valid.

2. We explored and evaluated a variety of data fusion techniques between image data and metadata. We identified methods where application proved beneficial and pinpointed weaknesses in the methodology.

3. We also proposed a method to aggregate various species into super groups, making them more distinguishable based on metadata.

4. We were able to demonstrate the potential for metadata to boost the performance of Wild Animal Classification models, a notable contribution to the field.

5. Our work also outlines effective strategies for incorporating metadata into WAC tasks, without increasing the expected annotation work done by experts.

Parts of this thesis has also been submitted and accepted to the the 11th European Workshop on Visual Information Processing. This conference paper can be read in Appendix E.

1.4 Use of Large Language Models

At the request of the consortium, this section is meant to highlight my usage of Large Language Models (LLMs) like ChatGPT[1]. The main way the tool has been used for me is in coding help, akin to how Stack Overflow has been used before, ChatGPT can take on a similar role. This helped speed up the process of learning new coding frameworks. Due to the cutoff date for ChatGPT, some information was inaccurate. In that case, documentation could be given to the model, and an updated answer was given, which was usually correct. If this still did not give the desired results, I often ended up reading the documentation myself.

Another application of LLMs was in the design of the report. Mainly, the help was given in the form of table generation, for more complex or multi-page tables. Queries to an LLM were faster than searching up and constructing the tables via online documentation.

LLMs were not used for literature search, as they are unreliable at conveying accurate information, but it can be hard to identify when this information is false. Since literature review involves acquiring new knowledge, one cannot easily hand this task over to an LLM without risk of false statements being added to the thesis work.

1.5 Thesis Structure

The paper is organized as follows. Chapter 2 presents essential concepts necessary for understanding this thesis. The reader may skip any or all topics in Chapter 2 if they have a strong understanding of the concepts discussed. Chapter 3 gives an overview of related work in the field of Camera Trap projects and Wild Animal Classification (WAC). Chapter 4 discusses how various data was acquired for our investigations into Wild Animal Classification (WAC), and how we created, modified, and used deep learning architectures. Chapter 5 gives the results of the methodologies applied during the thesis work, and discusses these results. Finally, Chapter 6 concludes our findings, and gives recommendations for further work in the field.

[1] `https://chat.openai.com/`

2 Background

This chapter will give the reader an understanding of the concepts used regularly in this thesis work. The chapter should give the reader all required knowledge to understand the concepts used throughout this paper. The reader is still encouraged to read the full papers referenced in this section if they wish to gain further insight into the tools used.

2.1 SCANDCAM Project

SCANDCAM is a project lead by Odden and Swenson (2023). The stated goal of the project is to: "Monitor medium and large mammals in the forest with the help of a large network of camera traps. The project is largely driven by local people from the Norwegian Hunting- and fishing association and other outdoor enthusiasts. The project also helps with data for monitoring of wild boar, lynx, and other animals. In addition, this data is utilized in several wildlife research projects that is conducted in cooperation with universities and colleges across Scandinavia"[1].

To collect this data, the project employs several types of cameras, mainly under the RECONYX brand[2], with various different models used. Some images were also taken with a BUSHNELL[3], no model was provided for BUSHNELL camera. Finally, the BROWNING, BTC-8FHD-PX[4] camera was used. The full table of cameras used can be seen in Table 2.1. This table also highlights the issue that we are lacking camera information from 62 thousand samples. Our work did not use camera type information as metadata, but other projects may struggle if they wish to utilize this information in their research.

[1] Disclaimer: This is paraphrased and translated from Norwegian by the author of this thesis
[2] `https://www.reconyx.com/`
[3] `https://www.bushnell.com/trail-cameras-2/`
[4] `https://browningtrailcameras.zendesk.com`

Table 2.1: *Full list of samples per camera type model*

Model	Samples
RECONYX, PC900 PROFESSIONAL	37557
RECONYX, HC500 HYPERFIRE	11657
RECONYX, PC800 PROFESSIONAL	7452
RECONYX, PC850 PROFESSIONAL	16972
RECONYX, HF2 PRO COVERT	16466
RECONYX, HC600 HYPERFIRE	5429
RECONYX, UltraFire	305
BROWNING, BTC-8FHD-PX	12054
RECONYX, HC500 HYPERFIRE	13
BUSHNELL,	343
RECONYX, SCANDLYNX	13
No information	62372

2.2 Deep learning for Image classification

Modern deep learning (DL) techniques enables a machine to solve abstract problems with astonishing accuracy given enough sample data Goodfellow et al. (2016) Krizhevsky et al. (2017). By giving a computer enough labeled data and a goal, a machine can learn to transform inputs into the desired output without further involvement by a human expert. This process is often conveyed as creating an estimator F such that $F(x) = y$, here x represents the input while y represents some output. Both the input and the output may be a value or a vector.

The pursuit of creating a good estimator purely from example data has been actively sought after for years, as not every problem in programming can be efficiently solved with algorithms by a programmer. However, Deep learning has not always been the go-to solution for problems where large amounts of labeled data is available.

This section will eventually build up to the current state of the art methods used in image classification using deep learning. However, to get there we first need to go back to the start of deep learning history.

2.2.1 Perceptron

The concept of the perceptron, the most important building block in a Deep Neural Network, was implemented back in 1957 Rosenblatt (1958). The mathematical background for this was created by McCulloch and Pitts (1943). The idea is to

combine a number of inputs and weights to give a strong enough signal to "activate" a neuron. A scalar bias (b) was also typically added to adjust how likely this neuron was to activate. The neuron had only two states, inactive or active, a binary classifier. Mathematically, the neuron can be described like this:

$$f(\mathcal{X}) = \begin{cases} 1 & \text{if } \mathcal{X} + b > 0, \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

where

$$\mathcal{X} = \sum_{i=1}^{n} w_i \cdot x_i \tag{2.2}$$

x_i here represents some scalar value while w_i is some scalar weight multiplied by the input value x_i.

This model came with two major issues: it failed to approximate some relatively simple mathematical operations. XOR is a typically given example, where you want your output to be one if either of the inputs are active, but not if both are active. Another issue was the lack of a differentiable equation.

This simple classifier was later combined into layers and sequences to form what we call a Multilayer Perceptron (MLP), which forms the basic concepts of the modern Deep Neural Nets.

2.2.2 Multilayer Perceptron

Along with his contributions to implementing the singular Perceptron, Rosenblatt also suggested a multilayer approach, where several pereceptrons were combined together to form a larger structure Rosenblatt (1958). It is important to note in this case, that a multilayer perceptron (MLP) is no better than a single layer of perceptrons unless some or all of the layers have non-linear activation functions. While a single layer of perceptrons cannot solve the XOR problem, a three-layer MLP can. (Goodfellow et al., 2016, pp. 171-177) has an excellent explanation for this, they use a two perceptron input layer, a two perceptron hidden layer h, and an output perceptron y. We can describe h and y as: $h = f^{(1)}(x; W, c)$ and $y = f^{(2)}(h; w, b)$. These equations also let us see why non-linearity is an important part of an MLP. Ignoring the bias, we could represent the transformation $h = W^\top x$ and $y = h^\top w$, which algebraically is equivalent to $y = w^\top W^\top x$. This simplifies down to $y = x^\top w'$ where $w' = Ww$. By using a nonlinear unit, Goodfellow (Goodfellow et al., 2016, pp. 171-177) create the non-linearity necessary to create an XOR expression. We can solve the XOR problem using these values: $b = 0$, $c = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$, $W = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$, and

$w = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$, which gives us:

$$X = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix} \implies XW \implies \begin{bmatrix} 0 & 0 \\ 1 & 1 \\ 1 & 1 \\ 2 & 2 \end{bmatrix} + c \implies$$
$$ReLU\left(\begin{bmatrix} 0 & -1 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix}\right) \implies \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix} w + b \implies \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \tag{2.3}$$

Equation (2.3) is explained in more detail in (Goodfellow et al., 2016, pp. 171 - 177).

The term MLP can be somewhat ambiguous, where some papers refer to the topic under the principles outlined in Rosenblatt (1958), based in McCulloch and Pitts (1943). While others may refer to MLP as any network with multiple neurons, connected by weights in a feedforward manner. Going forward, we will use the term deep neural network (or DNN for short) to encompass the general architecture of a feed-forward network taking n-dimensional inputs, passing them through hidden layers, and generating some m-dimensional output.

2.2.3 Loss Functions

A loss function aims to quantify how well some DNN performed. These metrics are especially important for updating the weights of the network, which is discussed in Section 2.2.4. One of the easiest conceptual loss functions is the Square Error function: $SE = \sum_{i=1}^{N}(f^*(x_i) - f(x_i))^2$, where $N = \#$ classes. This metric is quite easy to conceptualize. We measure a "distance" between the estimated function f^* and the actual function f, then we square that result to make sure all "distances" are summed together instead of canceling each other out. The larger the summation of distances, the worse the performance. While this loss function is quite easy to conceptualize, it is not that popular in deep learning fields. We will also not utilize this function, instead focusing on some other more popular functions, or functions better suited for our needs.

Categorical Cross-entropy

The cross-entropy of an estimated probability distribution q relative to the true distribution p, can in the discreet case be described as:

$$H(p,q) = -\sum_{i=1}^{N} p(x_i) log(q(x_i)) \tag{2.4}$$

Do note that in many binary classification problems, it is better to represent the loss function as:

$$H(p,q) = -p \cdot log(q) - (1-p) log(1-q) \tag{2.5}$$

Since we have a binary classifier p will either be 1 or 0, which means either the loss will be $-log(q)$ if the true class is 1, or $-log(1-q)$ if the true class is 0. this enables us to calculate cross entropy, without encountering issues with $log(0)$.

Oftentimes we want the loss value to be independent of the number of samples available. In this case we can modify the equation to divide by the total number of samples N:

$$H(p,q) = -\frac{1}{N} \sum_{i=1}^{N} p(x_i) log(q(x_i)) \tag{2.6}$$

An example may be helpful. Say we have a coin flip. Assuming the coin is fair, the distribution p should be $[0.5, 0.5]$, representing the probabilities of heads and tails respectively. We can then set our q distribution as $[0.5, 0.5]$ and slowly increase the bias towards tails and observe the increase in cross entropy as the bias increases from 0 (perfect predictor) to 0.5 (always tails). The cross entropy would increase as shown in Figure 2.1.

Focal Loss

Focal loss was introduced as a way to handle imbalanced datasets, where positive cases are obscured by a sea of negative cases. The method was proposed by Lin et al. (2018). The idea is to add a modulating factor γ to the cross entropy formula. They define the term p_t as follows:

$$p_t = \begin{cases} p & \text{if } y = 1, \\ (1-p) & \text{otherwise} \end{cases}$$

Where y is the true label and p is the predicted label. This makes the focal loss equation somewhat cleaner: $FL(p_t) = -(1-p_t)^{\gamma} log(p_t)$. The key idea is the inclusion of the focusing parameter γ. This focusing parameter is only included in the negative case side of the cross entropy loss equation (Equation (2.5)). Which in turn causes the loss function to devalue contributions from negative cases.

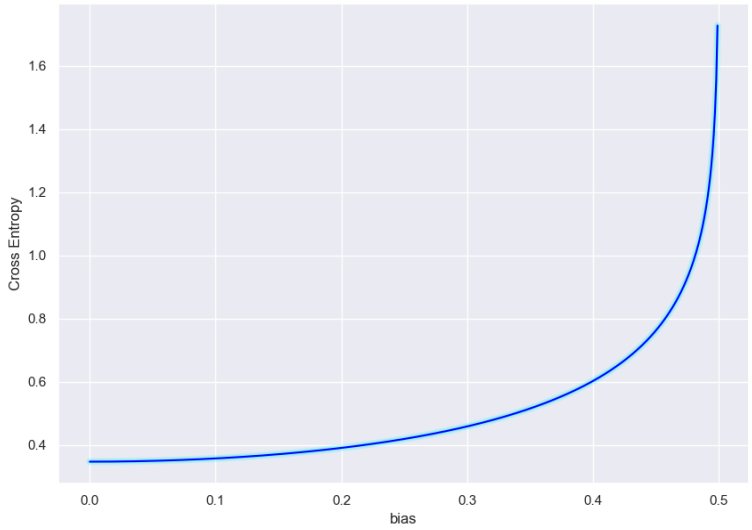


Figure 2.1: *Cross entropy of a coin flip predictor*

Categorical Focal Loss

Our thesis work focuses on multi class classification problems, which means we cannot use the binary version of focal loss in most cases. Luckily multi class versions of the focal loss has been developed. The cross entropy loss function used in deep learning is:

$$\frac{-1}{N}\sum_{i=1}^{N}[y_i \cdot log(\hat{y}_i) + (1 - y_i) \cdot log(\hat{y}_i)]$$

We add the focusing parameter as such:

$$\frac{-1}{N}\sum_{i=1}^{N}[y_i \cdot log(\hat{y}_i) + (1 - y_i)^{\gamma} \cdot log(\hat{y}_i)]$$

Which results in a generalized multi class focal loss function.

2.2.4 Back propagation

The previous example (Equation (2.3)) was fairly simple, containing only two input nodes, one output node and two hidden nodes in between. Working through this algebraic problem, while involved, is not infeasible to solve by hand. Real world applications of neural networks are rarely this simple. They usually involve several

hundreds of input values, combined in several hidden layers, and may result in a multi-dimensional continuous output variable.

This is where back-propagation, first proposed by Rumelhart et al. (1986), comes in. In normal operation, information only flows forward through the network. Back-propagation allows a network to learn by propagating the error backwards through the network. Using the gradient of the loss function, the network can adjust its weights to perform better on future samples. Its important to note that back-propagation simply refers to the calculation of the gradient. It is the role of the optimizer to actually update these weights (see Section 2.2.5 for further details on optimizers).

(Goodfellow et al., 2016, pp. 204 - 227) once again provides an excellent explanation of this procedure with examples. We can represent the output of the first hidden layer like this:

$$h^{(1)} = g^{(1)}(W^{(1)\top}x + b^{(1)})$$

In general we can represent the ith layer as:

$$h^{(i)} = g^{(i)}(W^{(i)\top}h^{(i-1)} + b^{(i-1)})$$

Here we represent $h^{(i)}$ as the ith layer in the structure. $g^{(i)}$ represents some function applied to the result, note that different layers may have different functions. $W^{(i)}$ is the weights of the ith layer, and $b^{(i)}$ represents the bias vector for the ith layer.

until we get to the final layer $\hat{y}$:

$$\hat{y} = g^{(N)}(W^{(N)\top}h^{(N-1)} + b^{(N-1)})$$

This result ($\hat{y}$) is then used together with the actual value y in supervised learning together with the loss function $\mathcal{L}$ to calculate the loss $\mathcal{L}(y, \hat{y})$. We want to find the gradient of this loss function with respect to every weight we have in our network. Working out the backprop algorithm we start with the partial derivative of the loss function with respect to the output of the final layer:

$$\frac{\partial \mathcal{L}}{\partial \hat{y}}$$

The partial derivative of the output of the final layer with respect to its input is:

$$\frac{\partial \hat{y}}{\partial h^{(N)}} = g^{(N)'}(W^{(N)\top}h^{(N-1)} + b^{(N-1)})$$

The partial derivative of the output of layer i+1 with respect to its input from layer i is:

$$\frac{\partial h^{(i+1)}}{\partial h^{(i)}} = g^{(i+1)'}(W^{(i+1)\top} h^{(i)} + b^{(i)}) \cdot W^{(i+1)}$$

The partial derivative of the output of layer i with respect to the weight from node j in layer i-1 to node k in layer i is:

$$\frac{\partial h^{(i)}}{\partial W^{(i)}_{jk}} = g^{(i)'}(W^{(i)\top} h^{(i-1)} + b^{(i-1)}) \cdot h^{(i-1)}_{j}$$

Using the chain rule, the partial derivative of the loss function with respect to a weight in the network is:

$$\frac{\partial \mathcal{L}}{\partial W^{(i)}_{jk}} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial h^{(N)}} \cdot \frac{\partial h^{(N)}}{\partial h^{(N-1)}} \cdot \ldots \cdot \frac{\partial h^{(i+1)}}{\partial h^{(i)}} \cdot \frac{\partial h^{(i)}}{\partial W^{(i)}_{jk}}$$

It is worth noting that to calculate the loss on layer $i-1$ we need the loss calculated at layer i. We can therefore speed up the algorithm significantly by storing the loss at each layer, then calculating the loss for the next layer in the back-propagation. This gradient can then be used to update the weights of the network using a variety of optimization algorithms.

2.2.5 Optimizers

Optimizers are the algorithms used to update the weights when the gradient is found. These range from the simple that subtract the gradient vector from the weights, to the complicated that include momentum, weight decay, variable learning rates, and more. All these parameters are included to try to nudge the network to a lower minima. If the hyperspace the loss function was situated in was convex and smooth, finding the global minima would be fairly straightforward. However, most real world cases for deep learning do not have a convex smooth space everywhere, and we need methods to nudge the network out of local minima, while preventing the network from overshooting once its nearing the global minima.

Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is the implementation of updating weights directly based on the gradient, with a few caveats. In most cases, deep learning algorithms do not use their entire dataset to calculate the gradient, instead it uses minibatches. The gradient is calculated for these minibatches instead of the entire dataset. Conceptually, we can imagine this as letting the network take more, but smaller steps that meander towards a minima, instead of fewer steps towards the

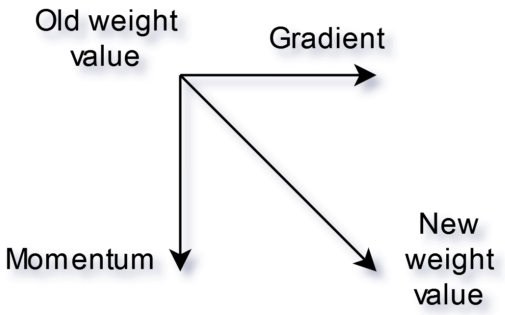


Figure 2.2: *Momentum based weight updates*

minima defined by all samples. We also include the learning rate γ. This parameter can be adjusted to finetune the step size of the weight updates. We can define it as:

$$W_{jk}^{(i)} = W_{jk}^{(i)} - \gamma \cdot \frac{\partial \mathcal{L}}{\partial W_{jk}^{(i)}}$$

The learning rate is there to prevent the network from overshooting the minima, resulting in a non-converging network.

We may also want to include a momentum parameter to this. In a way, momentum represents the direction the weight was traveling before the last calculated gradient. Many different variations of momentum have been proposed, and will be discussed further in the following sections. While exactly how momentum is calculated can be complex, the weight update after the fact is quite simple, as is shown in Figure 2.2.

ADAM

ADAM Kingma and Ba (2017) stands for "Adaptive Moment Estimation" and was a suggested improvement to the RMSProp algorithm, an unpublished algorithm proposed by Geoff Hinton[5], and AdaGrad Duchi et al. (2011). The update algorithm relies on a first moment vector $\boldsymbol{m}$ and $\boldsymbol{v}$ the gradient of the loss function $\mathcal{L}$. We also have two adjustable parameters β_1 and β_2. Initially $\boldsymbol{m}$ and $\boldsymbol{v}$ are zero vectors, but everything gets updated through the algorithm:

[5] `https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf`

$$
\begin{aligned}
m_t &= \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot \frac{\partial \mathcal{L}}{\partial W_{jk}^{(i)}} \\
v_t &= \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot \frac{\partial^2 \mathcal{L}}{(\partial W_{jk}^{(i)})^2} \\
\hat{m}_t &= \frac{m_t}{1-\beta_1^t} \\
\hat{v}_t &= \frac{v_t}{1-\beta_2^t} \\
W_{jk}^{(i)} &= W_{jk}^{(i)} - \gamma \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}
\end{aligned} \tag{2.7}
$$

the "time" t represent each run through the data. We also need to include a tiny number ϵ to prevent divide by zero issues.

LION: EvoLved Sign Momentum

The lion optimizer, proposed by Chen et al. (2023) is a modification of the ADAM family of optimizer. It is the optimizer used in the currently best performing models for the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). In their paper, they list out the pseudocode for the LION optimizer:

Require: $\beta_1, \beta_2, \lambda, \eta, f$
1: Initialize $\theta_0, m_0 \leftarrow 0$
2: **while** θ_t not converged **do**
3: $g_t \leftarrow \nabla_\theta f(\theta_{t-1})$
4: **update model parameters**
5: $c_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1) g_t$
6: $\theta_t \leftarrow \theta_{t-1} - \eta_t(\text{sign}(c_t) + \lambda\theta_{t-1})$
7: **update EMA of** g_t
8: $m_t \leftarrow \beta_2 m_{t-1} + (1-\beta_2) g_t$
9: **end while**
10: **return** θ_t

While this optimizer outperformed the standard ADAM for ILSVRC, it failed to give the same performance numbers in our tests. The optimizer was also slower than the implementation of ADAM. These factors made the optimizer rarely used in this thesis work.

(a) (b)

Figure 2.3: *Edge detection using Sobel Filters. Images by Dr. Xiyun Song*

2.2.6 Convolution

Convolution is a mathematical operator involving two functions f and g, creating a new function $f * g$. Since neural nets work in discreet values, we will focus on the discreet version of convolution. However, know that there is a formal definition for continuous functions f and g as well. The definition of a two dimensional convolution as used in deep learning is:

$$h = f * g = \sum_{i=0}^{N} \sum_{j=0}^{M} f(i,j) \cdot g(N-i, M-j) \tag{2.8}$$

This equation makes a couple of assumptions. The common definition defines convolution from $-\infty$ to ∞. However, we're working in image space, and rarely deal with images (represented by f here) of infinite size or resolution. The kernels (g) are also generally defined as smaller than the image size. Animations is an intuitive way to understand this process. If the reader wishes a demonstration of the operation, an example is provided by Dr. Xiyun Song[6]. Using specific values for the kernel, we can extract properties of the image. We can once again look at the examples given by Dr. Xiyun Song (Figure 2.3), using special kernels g, he extracted the edges of an image.

The motivation for convolution for image classification and DNNs relies on the idea that locality plays a role in images. This makes intuitive sense. If we see the head at the left middle side of an image, we expect a neck to be nearby, and not on the opposite side of the image. The benefit of introducing convolution is the reduced number of parameters. If we use the typical convolution kernel of 3×3, and an image of size 100×100, we would have to perform 9 multiplications per

[6] `https://coolgpu.github.io/coolgpu_blog/github/pages/2020/10/04/convolution.html`

kernel shift, 98 kernel shifts per row and a total of 98 rows. resulting in a total of $9 \times 98^2 = 86436$ operations. A significant number, but considering the same case with a fully connected network of 100^2 input nodes and 98^2 output nodes, with weights connecting everything means we have $100^2 * 98^2 = 96040000$ weights, three orders of magnitude larger. Factor in multiple layers and it becomes clear why convolution is favored in feature extraction. In real world applications we often want more than just one feature map generated from the convolution. We therefore use more than one kernel. It is still a significantly less computationally expensive process than using a fully connected network between layers.

2.2.7 Transfer Learning

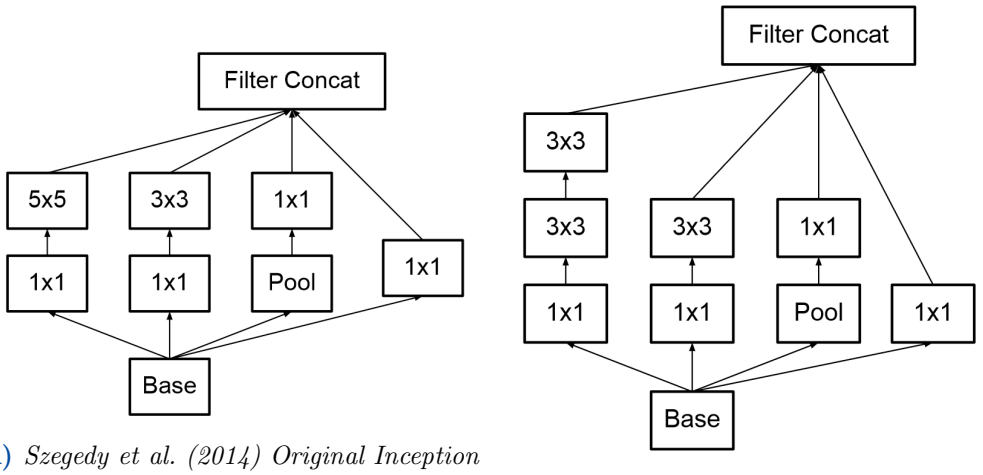
In many cases, we want a network to perform some specific task, but have a limited number of data points available. It can help to let the network learn more general features on a similar dataset first. This lets the network learn low level features shared between the datasets first. Then, by resetting the last few layers in a network and training only those layers, we can benefit from the feature extraction process a network has acquired, to get improved accuracy on our own, smaller, dataset. This phenomena has been detailed by several papers, many of which are included in the survey by Zhuang et al. (2020).

2.3 Existing models

No research happens in isolation. Several other groups of researchers have come up with full systems of deep neural networks. The networks discussed in this section have shown great promise, by coming up with novel workarounds for problems present in the deep learning scene, they have achieved greater accuracies on benchmark datasets than predecessors. We get to benefit from their research, as their models are publicly available for use by other parties.

2.3.1 Alex Net

AlexNet, proposed by Krizhevsky et al. (2012) is the oldest network this thesis looked at. It was among the first models used on large scale image datasets. They benefitted from several new innovations in the field, among them using the Rectified Linear Unit (ReLU) proposed by Nair and Hinton (2010) function as their activation function, arguing for its ease of calculating derivatives over the then more popular sigmoid ($\frac{1}{1+e^{-x}}$) and *tanh* function. To prevent overfitting they also employed label preserving image augmentation (more information in Section 2.5.3), specifically



(a) *Szegedy et al. (2014) Original Inception Module*

(b) *Szegedy et al. (2015) Inception v3 block*

Figure 2.4: *Original and v3 Inception block*

image translation, image flipping, and color jitter. Furthermore, they included dropout layers in their architecture, which combats overfitting by disincentives a network from overly relying on a few nodes. The method of dropout was new at the time the network was proposed, being published by Hinton et al. (2012) the same year as AlexNet.

2.3.2 Inception v3

Inception v3 is a convolutional block used in the GoogLeNet family of networks by Szegedy et al. (2014). The original paper introduced the inception module (Figure 2.4a), which attempted to downsample and parallelize the processing to speed up training. By using different size kernels, the aim was to detect features at different scales. The original network also employed multiple endpoints for classification, allowing for better propagation of the gradient, limiting the effects of vanishing/exploding gradient.

Version three of InceptionNet employs several improvements to the original architecture. Firstly, instead of using differently sized convolutions, they used successive 3×3 convolutions. They hypothesized that successive 3×3 convolutions lost little information over larger convolutions. The smaller convolutions would also promote faster learning. The updated architecture for Inception v3 can be seen in Figure 2.4b.

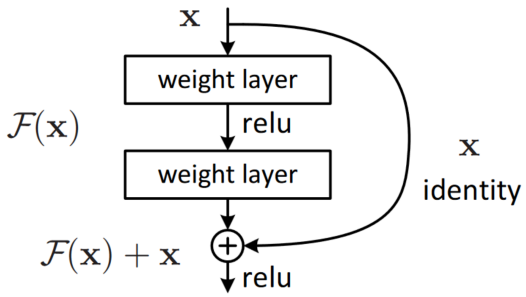


Figure 2.5: *He et al. (2016) representation of a skip connection*

2.3.3 ResNet

In deep learning, a relation was found between the number of layers and the performance of a network. There was however an issue with this methodology, as the number of layers increased, the training error of networks increased. This indicates that more than just overfitting is being introduced into the network. The solution proposed by He et al. (2016) is to allow an identity mapping between the input of a convolutional block and its output. Expressing the output of a block as $\mathcal{F}(\mathtt{x}) + \mathtt{x}$ instead of just $\mathcal{F}(\mathtt{x})$ allows the network to disregard the additions of a convolutional block, if it does not provide additional information for the network. The diagram given by He et al. (2016) shows this mapping efficiently (Figure 2.5).

2.3.4 EfficientNet

EfficientNet Tan and Le (2020) came out as a proposed method of finding more efficient ways of adjusting a networks depth d, feature map width w, and resolution r. They mainly propose that the depth, width and resolution should be scaled in a constant way following

$$\begin{aligned} d &= \alpha^{\phi} \\ w &= \beta^{\phi} \\ r &= \gamma^{\phi} \\ \text{s. t. } \alpha \cdot \beta^2 \cdot \gamma^2 &\approx 2 \\ \alpha \geq 1, \beta &\geq 1, \gamma \geq 1 \end{aligned} \tag{2.9}$$

Their experiments found $\alpha = 1.2$, $\beta = 1.1$, and $\gamma = 1.15$. They found these values using small scale networks, as searching on larger networks becomes prohibitively expensive. They then used the same values as they scaled up the network size.

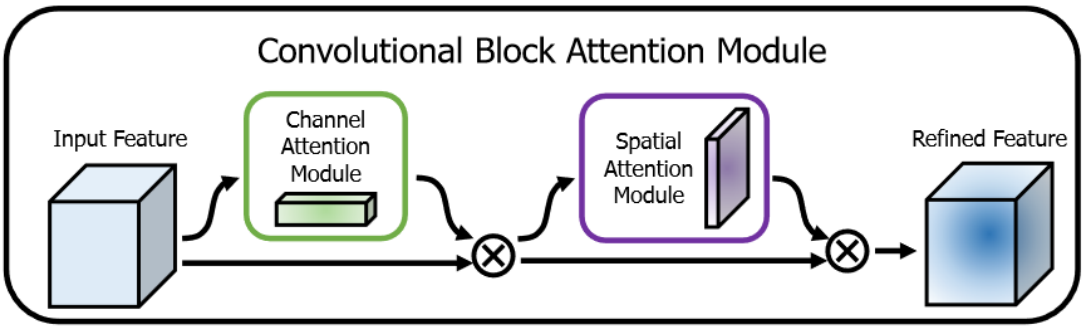


Figure 2.6: *Woo et al. (2018) CBAM architecture*

2.3.5 Channel Block Attention Module

The final, and newest, network architecture this thesis will look at is the Channel Block Attention Module (CBAM). CBAM, proposed by Woo et al. (2018). Before we can discuss the details, we first need to briefly introduce attention mechanics.

Attention, refers in deep learning to a network focusing more on certain parts of the input data over other parts. This can be beneficial in image processing, where we may have a large image, but only part of the image contains our subject. Imagine a picture containing a mouse, most of the image would only contain background information, irrelevant to the actual classification of a mouse. We want the network to focus on the region of the image where the rat is contained, and disregard the rest.

At a high level, the attention mechanism employed by Woo et al. (2018) is easy to see in Figure 2.6. Here we multiply the incoming feature map by a channel attention vector and a block attention module. Mathematically written as:

$$\begin{aligned} F' &= M_c(F) \otimes F, \\ F'' &= M_s(F') \otimes F' \end{aligned} \tag{2.10}$$

Where $\otimes$ represents an element wise multiplication.

This view glosses over exactly how M_c and M_s are found. Taking the mathematical derivations from Woo et al. (2018) we see that M_c is defined as:

$$M_c(F) = \sigma(MLP(AvgPool(F)) + MLP(MaxPool(f)) \tag{2.11}$$

and the block attention map M_s is found like so:

$$M_s(F) = \sigma(f^{7\times7}([AvgPool(F); MaxPool(F)])) \tag{2.12}$$

Here σ is shorthand for the sigmoid function, and $f^{7\times7}$ denotes a convolution of the input by a 7 by 7 kernel.

2.4 Data Fusion

Data fusion is the process of combining different types of data to give a fuller description of some problem. The goal of this extra context in our case is the improved prediction performance of deep learning models for WAC.

Bhatt and Kankanhalli (2011) defines two broad categories of data fusion: Late fusion, also known as decision-level fusion. This approach is generally regarded as easier to implement, but also fails to assist in feature extraction augmented by the extra data available. The other method, feature fusion or early fusion, aims to extract better features from the original data.

In deep learning for image recognition, feature fusion would use the different types of data during convolution, creating a new and better feature vector. While late fusion would concatenate the extra data to the feature vector generated by a convolution network.

2.5 Data Augmentation

Data augmentation in this setting is the method of increasing the dataset artificially, using label preserving transformations. Before we discuss the augmentation techniques used in this thesis, let us first discuss the motivation for augmentation.

2.5.1 Class Imbalance

In most real world cases, the distribution between classes is not even. This can cause problems for artificial intelligence (AI) and deep learning (DL) algorithms. The optimization problem often becomes more focused on predicting the majority class, instead of finding the underlying pattern in the data. Predicting the majority class is fairly simple compared to actually finding patterns in data, so we need to discurage our algorithm from finding the majority class. Shorten and Khoshgoftaar (2019) gives a recent and comprehensive survey on the methods and benefits of data augmentation for increased performance on minority class prediction. Our dataset is no different, with a few of the class labels representing the majority of the dataset, we need to employ techniques to mask this during network training.

2.5.2 Synthetic Minority Over-sampling Technique

Synthetic Minority Over-sampling Technique, or SMOTE for short, is an algorithm proposed by Chawla et al. (2002). Their proposed algorithm has been shown to improve detection of minority class instances.

However, the SMOTE algorithm also has a weakness, in that it may generate synthetic samples anywhere in the higher dimensional space. Borderline SMOTE proposed by Han et al. (2005) aims to improve this. By only generating synthetic samples on the boundary region between classes, the network gets more hard to tell samples, which should provide more benefit during training. Borderline SMOTE takes the set of all samples in the minority class class $P = \{p_1, p_2, \ldots, p_{pnum}\}$ and majority class $N = \{n_1, n_2, \ldots, n_{nnum}\}$. The next step is to count the number of majority class samples among the m nearest neighbors of a given point p_i. This number m' could be any number between $0 \leq m' \leq m$. If $m' = m$ the point is marked as noise and ignored going forward. If $m/2 \leq m' \leq m$ the minority sample is marked as a "DANGER" point, as it is likely to be misclassified, as 50% or more of its nearby neighbors are of the majority class. After all minority samples are checked, new synthetic samples are generated based on the minority samples in the DANGER group.

2.5.3 Image Augmentation

Image augmentation techniques are the specific methods we can use to alter an image while preserving the label associated with the image. The label in our case is the animal present in the image. Since our dataset contains sparse label information, there are a few augmentation techniques we cannot perform. Cropping or translating the image, risks moving the target outside the image frame. This can be alleviated if a bounding box is given for the animal. But as no such bounding box is given for our dataset we have to avoid transformations that remove a significant amount of the pixels in the original image.

We will use a couple notation tricks when talking about image augmentation going forward. They may be useful to note now:

- I: The original image.
- Ω: The modified image.
- i, j: Variable index of an image. The value of $I_{i,j}$ is assumed to be scalar.
- M, N: The size of the image: $0 \leq i < M$, $0 \leq j < N$

While these samples assume grayscale image, one can easily transfer the principles to color images by applying the same transformation across all color channels.

Image Rotation

Image rotation can be defined using transformation matricies. Given an image I we have a pixel value for each $I_{i,j}$, We can then define the new image $\Omega_{i',j'}$ using

the rotation matrix M:

$$M = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.13}$$

We find i' and j' using:

$$\begin{bmatrix} i' \\ j' \\ 1 \end{bmatrix} = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} i \\ j \\ 1 \end{bmatrix} \tag{2.14}$$

In reality, it is often easier to use the inverse matrix M^{-1}, as this gives us the location in the original image to look for the pixel value to palce in Ω. This also enables easier interpolation when

$$M^{-1} \cdot \begin{bmatrix} i' \\ j' \\ 1 \end{bmatrix}$$

gives us a fractional value for i and j. Many interpolation algorithms exists to infer the value for these fractional locations, we used bilinear interpolation.

Image Flipping

Image flipping refers to the process of taking either mirroring the image along the horizontal or vertical image plane. Mathematically expressed as:

$$\Omega_{i,N-j-1} = I_{i,j}, \text{ or } \Omega_{M-i-1,j} = I_{i,j} \tag{2.15}$$

A single flip will generate a distinct image compared to rotation. However, flipping the image in both the horizontal and vertical direction will produce the same result as image rotation of 180°. This thesis only utilized horizontal image flipping, as this produces images that are more likely to occur when taking images of animals using camera traps. We do not expect animals to appear upside down, with the ground located above them, when using camera traps.

Color Jitter

Color jitter operates on these four values: Hue, saturation, brightness and contrast. Color jitter is performed by first converting our RGB image to corresponding Hue, saturation, brightness (HSB) image. Calculating HSB from RGB can be somewhat involved, and is therefore omitted from this paper as it is only tangentially relevant

to the thesis. But simplified we can think of Hue as the Color "angle", where red is defined as 0°. Saturation describes how much of that hue we have from 0 (grayscale) to 1. Brightness is how light the color is, a lower brightness leads to darker colors, while maximum brightness would result in a white color. The final parameter adjusted with color jitter is contrast. Contrast is a measure of change between pixels ($\frac{\partial I}{\partial i}$, or $\frac{\partial I}{\partial j}$). To reduce the contrast we can reduce the average grayscale value:

$$\Omega_{i,j} = k \cdot I_{i,j}, \text{ where } 0 < k < 1$$

Enhancement can be a bit more involved, but in general is performed by increasing the distance between the lowest grayscale value and the highest.

Contrast reduction is generally regarded as easier, as due to the discreet nature of images, information is often lost when reducing the contrast of an image. This information is difficult to reconstruct with an inverse contrast enhancement algorithm.

Cutout

DeVries and Taylor (2017) suggests another method of label preserving image augmentation. The algorithm blacks out small patches of the original image before they are given to the network, forcing the network to become more robust against noise.

2.6 Dimensionality Reduction

The metadata collected in this thesis work is of a quite high dimensionality. We wished to utilize dimensionality reduction algorithms in order to represent the data in some way that is visually processable. We made use of the novel Uniform Manifold Approximation and Projection (UMAP) by McInnes et al. (2020). This algorithm based on higher dimension manifolds and topology. McInnes et al. (2020) demonstrates the qualitative and quantitative superiority of UMAP over several other dimensionality reduction algorithms like t-SNE, LargeVis, Laplacian Eigenmaps, and PCA. The underlying math of the method relies on a good understanding of topology. However, the actual algorithm can be understood with a few axioms assumed to be true:

1. There exists a manifold on which the data would be uniformly distributed.
2. The underlying manifold of interest is locally connected.
3. Preserving the topological structure of this manifold is the primary goal

Given these axioms we can perform a number of steps to create the manifold:

1. Create a weighted k-neighbor graph, weighted by some distance metric d.
2. foreach x_i find ρ_i and σ_i (Equations 2.16 and 2.18).
3. Create a weighted directed graph. The vertices is represented by the dataset. The directed edges and weights are outgoing from x_i to all x_{i_j} with weights given by Equation (2.17).
4. We can represent these weights as an adjacency matrix A. A is used in B as $B = A + A^\top - A \circ A^\top$. The researchers describe $\circ$ as "Hadamard (or pointwise) product".
5. Repeating this step for all x_i gives us the undirected graph G.

$$\rho_i = \min\{d(x_i, x_{i_j}) \mid 1 \leq j \leq k, d(x_i, x_{i_j}) > 0\} \tag{2.16}$$

$$w = \frac{-max(0, d(x_i, x_{i_j}) - \rho_i)}{\sigma_i} \tag{2.17}$$

$$\sum_{j=1}^{k} exp(w) = log_2(k) \tag{2.18}$$

k is a given hyperparameter. $d(x_i, x_{i_j})$ represents the distance between x_i and x_{i_j}. σ_i is implicitly defined to satisfy Equation (2.18).

Once the graph is created, we can reduce the dimensionality of the graph to our desired dimension. This is done by minimizing the cross entropy between the graph G and its lower dimension representation graph H. H consists of a set of points $Y = \{y_1, y_2, \ldots, y_N\}$ in our desired dimension. To group these points, we apply a set of attracting and repelling forces. The attractive forces between two vertices are given by Equation (2.19), and for each time an attractive force is applied to an edge, one of the vertices connected to that edge is repelled according to Equation (2.20). In these equations, a and b are set hyper-parameters.

$$\frac{-2ab||y_i - y_j||_2^{2(b-1)}}{1 + ||y_i - y_j||_2^2} w(x_i, x_j)(y_i - y_j) \tag{2.19}$$

$$\frac{2b}{(\epsilon + ||y_i - y_j||_2^2)(1 + a||y_i - y_j||_2^{2b}} (1 - w(x_i, x_j))(y_i - y_j) \tag{2.20}$$

This explanation is quite involved, but an excellent summary was given by Afridi et al. (2022). They break down the process into two major steps and a couple minor steps in each major step as so:

1 Learn manifold structure

  1.1 Finding nearest neighbors

  1.2 Constructing neighbors graph

    1.2.1 Varying distance

    1.2.2 Local connectivity

    1.2.3 Fuzzy area

    1.2.4 Merging of edges

2 Finding low-dimensional representation

  2.1 Minimum distance

  2.2 Minimizing the cost function

2.7 Evaluation Metrics

These metrics are typically used when assessing the performance of a machine learning algorithm. They are generally found by taking away part of the dataset as "unseen" samples. By running the algorithms on the input and comparing the models result against the true labels, we can derive different performance metrics for the network.

To discuss these performance metrics, a few variables can be useful:

- TP: True positive, the number of samples correctly placed in a class.
- TN: True negative, the number of samples correctly not placed in a class.
- FP: False positives, the number of samples wrongly placed in a class.
- FN: False negative, the number of samples wrongly not placed in a class.

These metrics can be conceptualized easily in a binary class:

$$\begin{matrix} & Prediction \\ Actual & \begin{bmatrix} TN & FP \\ FN & TP \end{bmatrix} \end{matrix} \tag{2.21}$$

For a multi class problem, the matrix becomes a bit more involved, but can be more generally summed up as:

$$\begin{array}{c} \\ Actual \end{array} \begin{array}{c} \begin{array}{ccccccc} & & & & Prediciton & & \end{array} \\ \begin{bmatrix} TN & \dots & TN & FP & TN & \dots & TN \\ \vdots & \ddots & TN & \vdots & TN & \iddots & TN \\ TN & \dots & TN & FP & TN & \dots & TN \\ FN & \dots & FN & TP & FN & \dots & FN \\ TN & \dots & TN & FP & TN & \dots & TN \\ \vdots & \iddots & TN & \vdots & TN & \ddots & TN \\ TN & \dots & TN & FP & TN & \dots & TN \end{bmatrix} \end{array} \tag{2.22}$$

In essence, everything not in the row or column of a specific class is a true negative. Everything in the column of the class but not the row of the class is a false negative (wrong class predicted), and everything in the row of the class but not the column is a false positive (class predicted wrongly). This pattern holds for all classes in a multi class problem.

2.7.1 Metrics used

This thesis will use some commonly used metrics: Accuracy, precision, recall, F_1 score, false positive rate (FPR), and false negative rate (FNR). The formal definition for these are:

$$\begin{aligned} Accuracy &= \frac{TP + TN}{TP + TN + FP + FN} \\ Precision &= \frac{TP}{TP + FP} \\ Recall &= \frac{TP}{TP + FN} \\ F_1 &= \frac{2TP}{2TP + FP + FN} \\ FPR &= \frac{FP}{FP + TN} \\ FNR &= \frac{FN}{FN + TP} \end{aligned}$$

Accuracy here is the individual class accuracy, and not the overall accuracy of the model. In that case we instead use:

$$Overall\ Accuracy = \frac{1}{N} \sum_{i=0}^{N} k_i$$

where

$$k_i = \begin{cases} 1 & \text{if } \hat{y}_i = y_i, \\ 0 & \text{otherwise} \end{cases}$$

Finally, because of the imbalanced nature of our dataset, it can be useful to include a metric sensitive to prediction accuracy accounting for class imbalance. We will be using Cohens kappa score, proposed by Cohen (1960).

2.7.2 Cohen Kappa Score

Cohen kappa score measures the agreement between two predictors who classify N items into C mutually exclusive classes. To find this agreement we need to first find the probability of our two predictors predicting identically by random chance p_e:

$$p_e = \frac{1}{N^2} \sum_{k=1}^{C} n_k^{(1)} n_k^{(2)}$$

Where $n_k^{(i)}$ is the number of times predictor i predicted class k.

P_o is the observed agreement between samples. Given some observed response matrix M:

$$M = \begin{bmatrix} x_{1,1} & \dots & x_{1,C} \\ \vdots & \ddots & \vdots \\ x_{C,1} & \dots & x_{C,C} \end{bmatrix}$$

p_o is given as:

$$p_o = \frac{\sum_{i=1}^{C} x_{i,i}}{\sum_{i=0}^{C} \sum_{j=1}^{C} x_{i,j}}$$

Finally, we can use p_e and p_o to find the overall kappa score:

$$\kappa = \frac{p_o - p_e}{1 - p_e}$$

2.7.3 Intersection Over Union

Intersection over Union (IoU) or Jaccard index is not directly used in this thesis, but is discussed in some of the related work. For formality's sake, we have included it here. The definition of IoU is:

$$IoU = \frac{A \cap B}{A \cup B}$$

Visually, it can be represented as shown in Figure 2.7. This metric is often used when bounding boxes or location masks are used for deep learning and image processing. It is a convenient metric for finding the overlap between a predicted bounding box or truth mask, and the actual bounding box or truth mask.

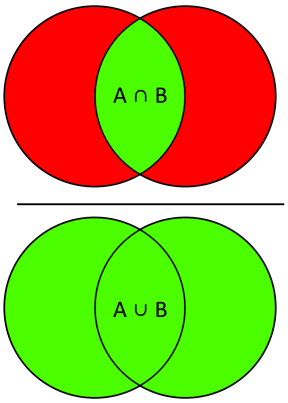


Figure 2.7: *Intersection over Union*

2.7.4 Micro versus Macro average

When looking at a multi-class problem, we can talk about metrics respective to each class. Accuracy, precision, recall, FPR, and FNR are discussed in this thesis. To get an overview of the performance of models, it can often be more efficient to look at the overall performance for all classes combined.

However, averaging the numbers poses an important question. Should the average account for the frequency of each class. This is the central question when performing micro or macro averaging.

An example is helpful to demonstrate. Calculating the recall for a 3 class predictor where:

- Class 1: 80 TP and 20 FN
- Class 2: 80 TP and 20 FN
- Class 3: 2 TP and 8 FN

The macro average would be $\frac{0.8+0.8+0.2}{3} = 0.6$. While the micro average would be $\frac{80+80+2}{100+100+10} = 0.77$. The micro average here gives a more true representation

of the actual performance of the predictor, accounting for the class imbalance. However, neural networks are in general already prone to favor majority classes over minority classes. Ecologists, also generally care more about the minority classes, as the minority classes are generally represented by animals that may be endangered. We will therefore mainly prioritize macro averaging when reporting results.

3 Related Work

The aim of this section is to briefly visit other related work in the field. We will outline the relevant factors when setting up a camera trap project. We will also highlight many of the existing camera trap projects with annotated data publicly available. Furthermore, we will discuss how these datasets have been used to develop State-of-the-art (SOTA) deep learning models for Wild Animal Classification (WAC). Finally, we will highlight some of the challenges with data fusion and multimodal image recognition.

3.1 On proper use of Camera Traps

Meek et al. (2014) and Falzon et al. (2019) goes over the principles and best practices for standardization of camera trap setups. The motivation of this thesis is to make comparisons easier.

The papers discusses a few different principles that need to be in order when reporting results. These are: camera model(s), mode of deployment, camera settings, environment, and study design.

Camera model(s) is quite straightforward, different models have different camera and motion sensors. This may affect when the camera triggers, and the resulting image.

Mode of deployment is in essence the methodology used when placing the camera traps; were the traps randomly placed, or in a regular grid? In the case of randomly placed camera traps, what is the region the traps were placed in? What were the minimum, maximum and average distance between traps? If the cameras were placed along a regular grid, what distance was there between each trap, and could the researchers place the trap within a certain range of this grid, or did they have to place it at exact geographical locations? Were traps placed in deliberately biased areas, where the researchers expect to see a certain kind of animal more often? If this is the case, it needs to be disclosed. Sometimes, the work also requires modifications of the habitat the camera is in, such as clearing branches for a clear view or assembling scaffolding the camera trap can be attached

to. If any modifications are performed of the habitat, these need to be disclosed as well. Finally, some trap setups will make use of bait in order to lure in animals. Lures need to be disclosed as well if they are used.

Camera settings should also be discussed. These are settings such as video versus still image. How long is the delay between sensed movement and image capture? If multiple images are captured, what is the delay between images? The camera placement with respect to the ground and orientation should also be given, as the height above ground level will affect what kinds of animals are detected. The orientation will also have an effect on how animals are detected.

Environment was also discussed as an important factor. Here, one of the most relevant cases is the IR noise generated by stray heat. A camera trap will perform worse if the temperature gradient between the animal and the background is small. The sensor may also then be affected by such factors as direct sunlight. If the sun heats up the entire sensor, the camera trap becomes less sensitive to movement by animals.

Finally, the number of images captured, and number of events, as well as the time period in which they were captured is also relevant information. Any paper using camera trap data should also mention number of false positive and false negative images, as well as number of broken and stolen camera traps.

To summarize the findings, a camera trap project should inform about:

- Camera Model(s): Different models have varying camera and motion sensors, affecting when the camera triggers and the resulting image quality.

- Mode of Deployment: How was the cameras set up, were they randomly placed, or set in a regular grid? Do they have some deliberate bias to their placement?

- Camera settings: These include whether video or still image is used, the delay between sensed movement and image capture, and the camera's placement and orientation.

- Study Design: Number of images captured, time period for image capture, and number of false positives and negatives should be reported. Stolen and broken camera traps should also be noted.

These principles ensure that the data collected is reliable and can be compared across different studies.

3.2 Animal Camera Trap Projects

To perform any kind of deep learning, a large dataset is required in order to tune the network properly (see Section 2.2, or just Section 2.2.4 and Section 2.2.5 for specifics). Luckily, there are several projects from several continents that capture and label camera trap images. We have tabulated a list of several of these, which can be seen in Table 3.1. A brief explanation of the table: the "Name" refers to the name of the project. We've also included a citation for the reader to trace the project more easily if desired. "Count" refers to the total number of images. This may include human images, which are generally removed from the public dataset due to privacy issues. "Cat." abbreviation for categories is the number of animal categories present. Most datasets have an empty category as well, this is removed from the category count. "B. Boxes" refers to the number of bounding boxes supplied with the dataset. Images may contain more than one bounding box. Lastly, "Empty" is the approximate percentage of images that contain no animals. This is a common occurrence in camera trap datasets, as researchers wish the cameras to be too sensitive to movement, rather than not sensitive enough.

3.3 SOTA Classification of WAC

Norouzzadeh et al. (2021) proposes a method not only for image classification, but also suggests that object detection improves the results of image classification. Their justification is that object detection helps remove irrelevant background information from the images, helping the network discard data not needed to determine the animal in the image. The authors also suggest that this performance increase come without the requirement of extra data needed. Their pipeline used an existing pre-trained model for object detection. Even so, they managed an accuracy of 91.71%, with a precision of 84.47% and a recall of 84.24%.

Since bounding boxes were already created at this step, the authors could fairly easily count the number of animals in each scene by counting the number of bounding boxes the model gave.

After images were separated into empty and non-empty images, the last step was to classify what kind of animal was in the non-empty images. Since the target dataset they used was heavily imbalanced, the model managed to classify the majority classes with a very high accuracy of 97.7% or better. The paper was less specific on the minority classes but achieved an overall accuracy of 91.37%.

The paper also discusses active learning methods, wherein the model is first fed some data to train on, then the model will find the most relevant unlabeled data points, which the model then can ask an oracle (typically a human) the ground

Table 3.1: *Camera Trap Projects*

Name	Count	Cat.	B. Boxes	Empty
Caltech Camera Traps Beery et al. (2018)	243K	21	66K	70%
ENA24 Yousif et al. (2019)	10K	23	All	0%
Missouri Camera Traps Zhang et al. (2016)	25K	20	900	0%
North American Camera Trap Images Tabak et al. (2019)	3.7M	28	8892	12%
WCS Camera Traps Wildlife Conservation Society (2019)	1.4M	675	375K	50%
Wellington Camera Traps Anton et al. (2018)	270K	15	0	17%
Island Conservation Camera Traps Island Conservation (2020)	123K	47	65K	60%
Channel Island Camera Traps The Nature Conservancy (2021)	247K	5	All	47%
Idaho Camera Traps Idaho Department of Fish and Game (2021)	1.5M	62	0	70.5%
Snapshot Serengeti Swanson et al. (2015)	2.65M	61	150K	76%
Snapshot Karoo Snapshot Karoo (2019)	38K	38	0	83%
Snapshot Kgalagadi Snapshot Kgalagadi (2019)	10K	31	0	76%
Snapshot Enonkishu Snapshot Enonkishu (2019)	29K	39	0	65%
Snapshot Camdeboo Snapshot Camdeboo (2019)	30K	43	0	44%
Snapshot Mountain Zebra Snapshot Mountain Zebra (2019)	73K	54	0	91%
Snapshot Kruger Snapshot Kruger (2019)	10K	46	0	62%
SWG Camera Traps SWG (2021)	2M	120	102K	13%
Orinoquía Camera Traps Vélez et al. (2023)	105K	51	0	20%

truth information about. By letting the model itself decide which data points are most important for future learning, one can dramatically reduce the number of required data points in order to create a good deep neural network model.

Norouzzadeh et al. (2018) is another paper concerning itself with animal classification, object counting and action recognition. Additionally, they attempted to classify whether children were present in the image. They proposed a multi-stage fusion network, and showed that it outperformed a single shot full classifier model. The paper lays out four main objectives they want to achieve: Task one is to detect which images contain animals and which do not, task two is to classify the species of the animal in the non-empty images, task three is to count the number of animals in each image, and task four is to add additional attributes to each animal. The research team tested several models for the binary classification task to determine if an image were non-empty or empty. In the end the VGG (Simonyan and Zisserman (2014)) network won out, giving an overall accuracy of 96.8% accuracy. Task two, also achieved high accuracy scores, with a top-1 accuracy of 94.9% and a top-5 accuracy of 99.1%. The models struggled more to count the number of animals in the scene. The researchers somewhat simplified the problem here by dividing the animal count into bins. The model was expected to accurately count the number of animals from one to ten. However, one bin was dedicated to small herds (11-50 animals), and another bin for large heards (51+ animals). The accuracy here was 62.8% for predicting the correct bin, and 83.6% for counting within one bin of the actual number. Action detection achieved an accuracy of 75.6%, a precision of 84.5%,and a recall of 80.9%.

Schindler and Steinhage (2021) proposes a two stage fusion network in order to both classify the animal, and determining what kind of actions the animal is performing. A mask region-based convolutional neural network (Mask R-CNN) was chosen as the classification model. This type of model is exceedingly powerful if performing well, giving pixel level classification information instead of just bounding boxes around the targets. The paper also discussed the use of temporal information, as the data available was videos instead of still images. However, the performance of the Mask R-CNN was still better, and was therefore used for the segmentation and classification objective.

The action recognition network always used the available temporal data, and was based on different variations of ResNet-18. The input to the model was $3 \times T \times H \times W$, where T represents the number of frames in the clip. The paper mentions a typical T value to either be 8 or 16. The 3 is there due to the 3 color channels of the image. H and W represent the height and width of the image respectively. There was also a SlowFast network proposed by Feichtenhofer et al. (2019). However, this network architecture both performed worse, and was slower to train.

Schindler and Steinhage (2021) also defines their own accuracy metrics for segmentation, the metric used by the paper is to determine how confident the network is in the bounding box or segmentation mask, along with the Intersection over Union of said bounding box or segmentation mask. If both the confidence and the Intersection over Union is above 0.5, as well as the class label being correct, then the authors describe the match as a True Positive. If the confidence score is above the threshold while the Intersection over Union is below the threshold, or the class label is wrong, the paper describes it as an False Positive. False Negative is then defined when the confidence score is below the required threshold. The authors did not discuss the True Negative rate. With these criteria in mind, the best segmentation method found by the researchers achieved an average precision of 63.8%. As for action detection, the models here managed a much more respectable 94.1% accuracy.

3.4 Data fusion for deep learning

The topic of data fusion and deep learning has been explored by several authors for a myriad of fields. The goal with this data fusion is in general to give the model additional context to improve the decision-making of said model. Arevalo et al. (2017) propses a specific multimodal unit to handle data fusion, which they call a Gated Multimodal Unit (GMU). Their specific case handles two types of multimodal data, in which case the contribution of each data type is determined by either σ or $1 - \sigma$, where σ is a hyperparameter learned during model training. Utilizing their GMU, they combine textual descriptors with an image to predict the genre of movies.

Data fusion was also used for skin lesion classification. Pacheco and Krohling (2021) and Li et al. (2020) both propose methods for fusing patient data with the image data to improve prediction accuracy. The general findings from both papers is that early fusion outperforms late fusion strategies. The papers suggest that metadata enhances the feature extracition process, that can then be used for better classification.

Finally, Bi et al. (2022) proposes user generated hintmaps combined with data fusion for improving skin lesion segmentation. After an initial stage of feature extraction, the positive and negative hintmaps are combined with the image in several Hyper Integration Modules to improve the segmentation results. While the direct model would not be adaptable to the image and metadata case which this paper studies. The contributions of Bi et al. (2022) are still worth considering when designing architectures to predict animals in camera traps.

3.5 Handling Multimodal Image Data

An issue with camera trap images is the wide variety of lighting conditions which may occur during its operation. This has lead most cameras to at least employ two cameras, one RGB image camera for recording animals during daytime, and one IR camera for operation during nighttime. This however poses an extra problem for a deep learning model to overcome, as it must now be expected to both classify grayscale images, and color images. One option is to simply convert all images to grayscale. This does not however quite solve the issue as the grayscale images produced by converting an RGB image to grayscale does not produce the same result as an image captured from an IR camera. Liu (2018) proposes a reference less method for converting IR images to daytime RGB images. The method used did however introduce a significant amount of noise as well, meaning it may cause more harm than good to use this methodology.

de Lima et al. (2022) Offers the inverse method of Liu (2018). Where instead of reconstructing RGB images from IR images, they instead attempt to estimate the IR response using a Generative Adversarial Network (GAN). This thesis seemed to show more promising results than Liu (2018), this may be due to the already close to NIR imaging the red channel has, in fact most camera sensors use an IR filter to remove IR radiation when capturing images. However, no code was provided, and recreating these results would be infeasible in the time available.

This thesis has to deal with this issue, as the images generated by camera traps are multimodal, meaning daytime images are taken with an RGB sensor, while the nighttime images are taken with an IR sensor. This forces the network to either be color and illuminant agnostic. It cannot be only color agnostic, as converting RGB images to grayscale will not result in the same kind of lightness for animals as an IR image of that same animal. It would have to be animal detection on the general shape and size of animals only.

3.6 Summary

We can summarize the related work as follows:

- Meek et al. (2014) and Falzon et al. (2019) suggests several guidelines when creating camera trap projects. They reccomend listing out: Camera Model(s), mode of deplyment, Camera Settings, and study design.
- Section 3.2 lists out camera trap projects by several teams and contributors. The project consists of millions of labeled images with thousands of bounding box samples.

- Several proposed networks have been used on different datasets. Norouzzadeh et al. (2018) and Norouzzadeh et al. (2021) focuses on object detection and counting, image classifications, and action recognition for the Snapshot Serengeti dataset. They achieve accuracies ranging from 81% to 97.7% on different task across the different papers. Schindler and Steinhage (2021) and Feichtenhofer et al. (2019) focused more on the use of image sequences or videos for object detection and action recognition. Schindler and Steinhage (2021) was the best performer here, and pushed action recognition up to a respectable 94.1%.

- The goal of data fusion, as discussed in this section, is to provide extra context for a network to make a decision. Several methods have been proposed for this task. Arevalo et al. (2017) proposed a special neural unit to handle feature fusion. While Pacheco and Krohling (2021) and Li et al. (2020) propose using metadata as vectors that can interact with feature maps to create improved feature vectors. Bi et al. (2022) also proposes a fusion network, using user generated hintmaps. They show an improved segmentation network for skin lesions.

- Finally, we have discussed some issues related to the multimodality of Camera Trap images. While methods were explored to solve this issue (Liu (2018), de Lima et al. (2022)), we found issues with applying these to our models. The general consensus among classification problems is simply to use a mixture of IR and RGB images, and rely on the networks to become color agnostic.

4 Materials and Methods

This chapter describes the methodology used to acquire the data used for experiments, along with the experimental design. The aim is to give the reader the required knowledge to reproduce results found in this thesis work. However, the data was acquired with the permission of NINA, and may not be easily accessible by third parties without explicit permission from NINA. If access to data is desired, contact the authors of this thesis.

4.1 Datasets

4.1.1 Caltech Camera Traps

The Caltech camera trap project is the least used dataset in this thesis. The details of the dataset were discussed in Section 3.2, with the statistics given in Table 3.1. This dataset was only used to test how our finished models reacted on empty images. This dataset was used, as it has an environment closer to a Nordic setting, being located in southwestern US over Tanzania. It would be more ideal to use a more northerly dataset. However, not all of these contain the desired data type "empty".

We fetched 200 random empty images from this dataset, pruned any corrupted images and any images that were mislabeled (human images were sometimes labeled empty). This resulted in 191 images that were empty, which we could then attempt to classify using our networks.

4.1.2 Snapshot Serengeti

The Snapshot Serengeti (SS) dataset by Swanson et al. (2015), is a set of camera trap image sequences taken from the Serengeti National Park in Tanzania. It consists of 2,65 Million images of wildlife. The dataset is available through two avenues, direct browser download, or download through the tools gsutil or azcopy. Downloading the files through the browser is not recommended, as many factors

may cause the download to be canceled. Instead, the recommended way is to use gsutil or azcopy. By specifying a URL one can download specific seasons, or the entire dataset via command line interface.

The metadata on the other hand, while large, is not too large not to download via browser if desired. They are small enough that the risk of disruptions on the connection is negligible.

The full information on how to install images from the SS dataset can be found on Lila Science web page[1].

4.1.3 NINA Viltkamera

Overview

This dataset consists of images of several different types of animals in the Norwegian climate. The images have mostly been captured in central Norway. The next few sections will talk about how exactly the dataset was acquired, however a brief overview can be seen in Figure 4.1.

Basic Metadata

Acquisition of the NINA Viltkamera dataset was a significantly more involved step. The original intention was to recive a data blob from NINA that could be downloaded. However, this proved to be challenging for them to create, and NINA eventually aggreed to allow a webscraper to extract the information. All images are technically publicly available through the links: `https://viltkamera.nina.no/Media/h1-h2-h3-h4-h5.jpg`, where h1 through h5 represents hash values. The hash values have no real pattern to them, at least as far as we could tell, so accessing them through a naive ascending request paradigm would not work. Just to give an idea of the number of potential URLs we can observe the hash values:

- h1: 16^8 values
- h2, h3, and h4: 16^4 values each.
- h4: 16^{12} values.

This gives us a total number of $16^{(8+3\cdot4+12)} = 16^{32}$ unique image values.

This mapping is a commonly used technique for different devices to create "unique" ID's without communicating with other devices. The mapping is known as a "universally unique identifier" (UUID) and was introduced by Leach et al.

[1] `https://lila.science/image-access`

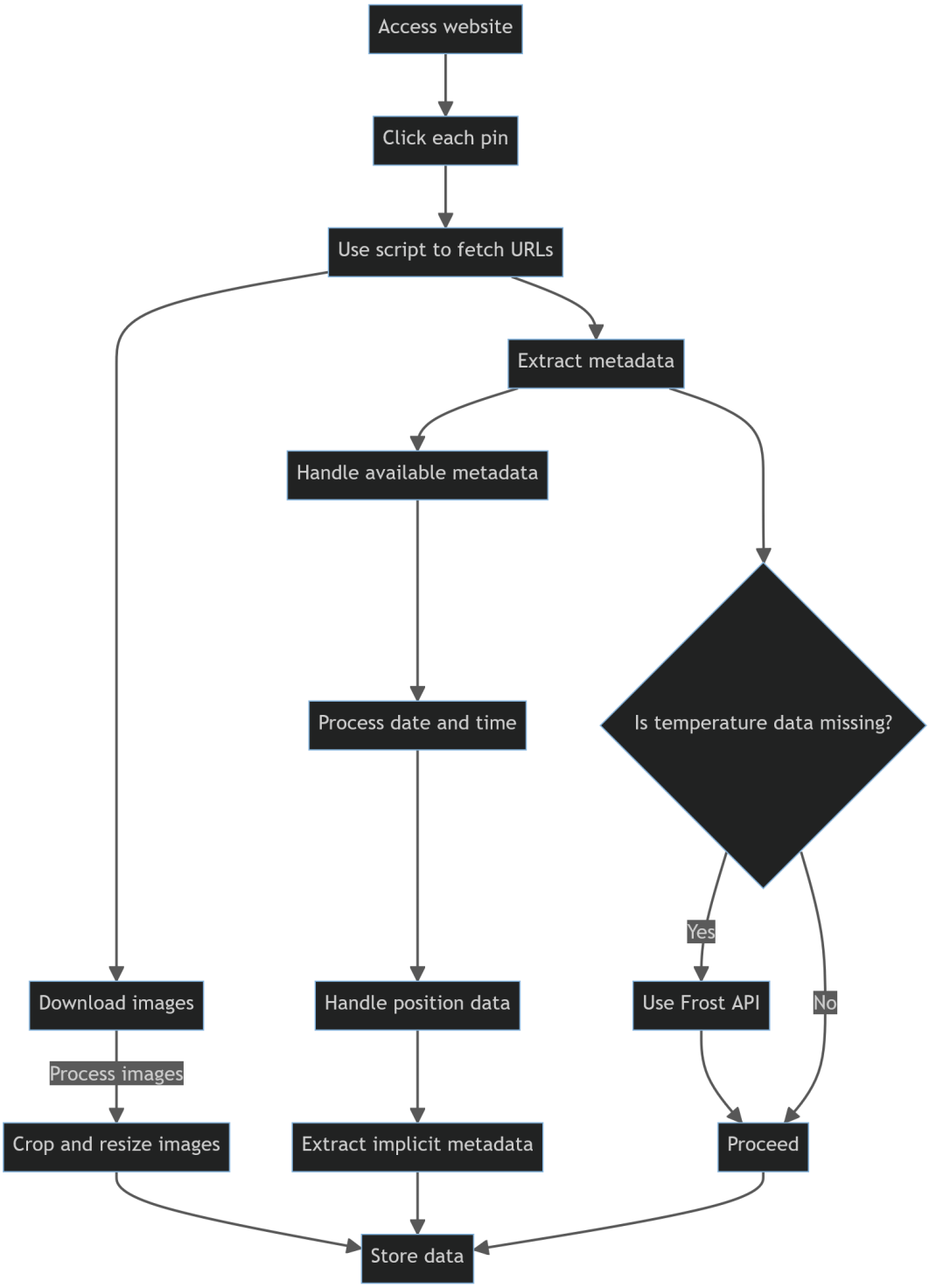


Figure 4.1: *Process for acquiring the NINA dataset*

(2005). This high probability of uniqueness means that it is completely infeasible to attempt a structured or exhaustive search of the possible UUID values.

Luckily, there is another solution to our problem. NINA has provided an interface website for accessing all the datapoints[2]. This website offers an interactive map with several pins (Figure 4.2). Each pin represents one camera trap, and clicking this pin lets us view the images taken with this camera trap (Figure 4.3).

Since the images are available through the website `https://viltkamera.nina.no/`, it means we can create a script to extract these image URLs as well.

To do this, we first had to identify where the URLs were stored. This was in an object on the website simply called "vm". It turns out, every time the user clicked a pin on the website, "vm" updated with the specific information of that pin. Furthermore, "vm" contained a public data member called "media". Each entry in "vm.media" contained a JSON object listing a filename (NOR: "Filnavn"), as well as a foreign key referencing the species id (NOR: "FK_ArtID"). The entry also contained various useful metadata to store. However, location info and the name of the species was missing here.

To access the reference between "FK_ArtID" and the species name, we had to utilize the function "vm.arter()". This returned a list of JSON objects with a key number "ArtID", and the corresponding name (e.g. "Bever"). In the same vein, we could find the approximate location of the camera by utilizing "vm.lokaliteter()". This function returned a Location ID, and its corresponding latitude and longitude. The location ID we knew from clicking on the specific pin, so it was a simple matching problem to find a latitude and longitude for a specific pin number, observing Figure 4.3 we see this specific camera had Location ID "4725".

The aforementioned steps could then be repeated for each pin present on the map to access all samples, and their corresponding image URL. Performing these steps for all pins would become tedious quickly, so instead we developed a compact script to be run in the browser to automatically fetch this information and download it as a JSON object. The full code can be found in Appendix B. Once the partial metadata had been acquired, we could continue by fetching image data, as well as fetch auxiliary metadata and filling missing metadata values.

4.1.4 Image Data

Once filenames were acquired, we could move on to downloading the images directly from the URL: `https://viltkamera.nina.no/Media/''filename''`. To minimize strain on NINAs servers, we throttled the download to a conservative 5 images per second. To speed up the process, we used the downtime between

[2] `https://viltkamera.nina.no/`

Figure 4.2: *Interactive map presented on* `https://viltkamera.nina.no/`

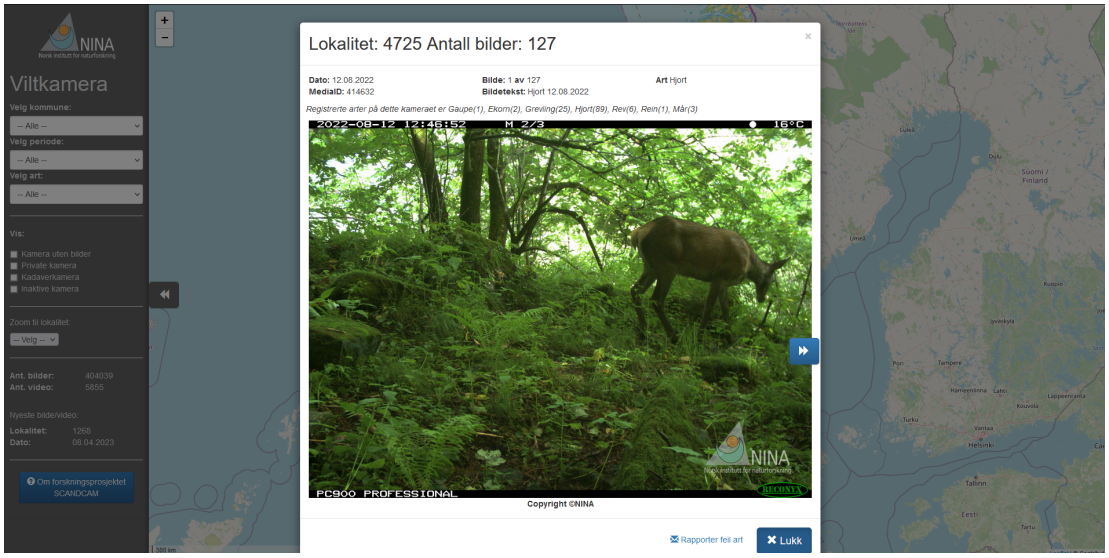


Figure 4.3: *Window after clicking a pin on* `https://viltkamera.nina.no/`

image downloads to process the images, extracting implicit metadata, as well as requesting other servers for missing temperature metadata.

Processing involved cropping out the bands of information at the top and bottom of each image. Afterwards, the image was resized to 512×512. A shape small enough to be easily portable, yet large enough that we would not struggle with low resolution in the future.

4.1.5 Metadata

Accessing metadata was also an involved process. Some of the metadata was included in the base file we could download from `https://viltkamera.nina.no`, either directly or indirectly by finding Foreign key links. However, temperature data was oftentimes incomplete here. Of the 170 thousand samples collected, over 90 thousand samples had missing temperature data. To recover this data we had to utilize other services that could give us an estimated temperature for the time and location of the datapoint. Luckily, no datapoints were missing location info or date information.

Temperature data

To estimate the missing temperature we utilized the Norwegian Metrological Institutes Frost API[3]. We used existing information about when an image was

[3] `https://frost.met.no/index.html`

taken, along with the latitude and longitude data to fetch temperature data from the closest weather station to the given camera trap. We limited the data point to ±24 hours of the time the image was captured. This may still have caused some issues, as not every image in the dataset had a corresponding weather station reading within 24 hours of the capture, nor does it account for natural temperature swings over a day.

To handle any values still missing after using the Frost API, we stored the temperature as a 2-dimensional vector. The first of these values told us whether the current temperature value was valid or not. If the value was 1, the following value was valid, if the first value was 0, the second value was not valid. This method was used to help tell the network when the temperature value could be trusted and when it should be ignored.

Datetime

The date and time values were stored as a one-hot encoded 67-dimensional vector. This was done due to the challenges of representing the circular nature of time to a neural network. By representing the month as a 12-dimensional vector, the day as a 31-dimensional vector, and the hour as a 24-dimensional vector, we could perfectly reconstruct the initial date while preventing discontinuities at the end of the year. We did consider using sine curves instead, since they would capture the circular nature better than a linear scale. However, we worried about the equivalent values which would occur around either seasons (spring and fall values would be identical), as well as the time of day (dawn and dusk would also be indistinguishable). This issue can be seen in Figure 4.4.

Position

The motivation for including latitude and longitude was the suspicion that different animals may roam different regions, especially if we included time into the picture as well. As mentioned before, the latitude and longitude was available through the web scraper. However, this information is somewhat imprecise, because the accuracy of the information is only given to about a kilometer's radius of the pins given location. We do not believe this approximate location information will be to the detriment of classification, as animals tend to roam in areas larger than 1 kilometer. We still believe it is worth disclosing the fact that we only have approximate location information, rather than precise location information.

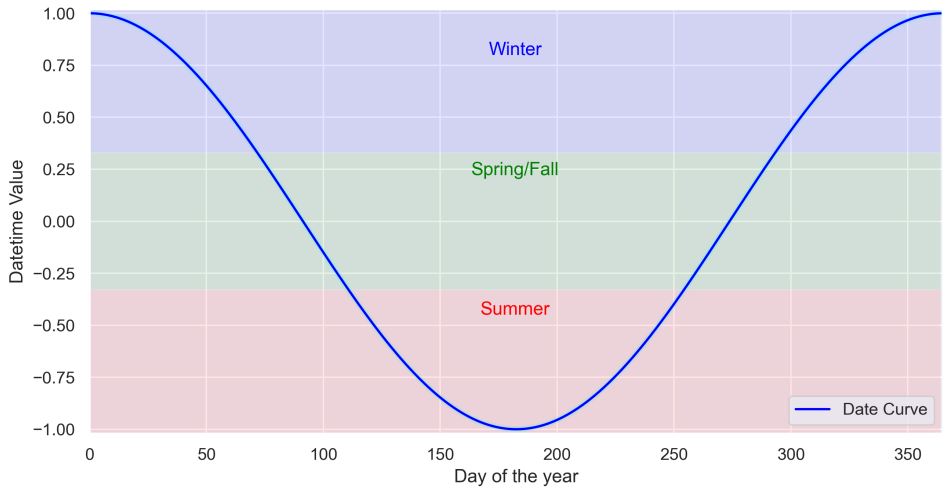


Figure 4.4: *Conversion of the day of the year to a sine curve.*

Implicit Metadata

Scene Attributes is the first implicit data that was fetched out using other deep learning models. Using the models outlined in Zhou et al. (2017), 102 scene attributes were extracted from each image, which was then stored with other metadata for that image. The labels associated with the scene attributes can be seen in Appendix C. These attributes could then later be fetched during our own model training. Finding the implicit metadata and storing it caused our own models to train faster, as we did not have to run another model mid training to get the required implicit metadata.

Along with scene attributes, the models pretrained on the Places365 dataset (Zhou et al. (2017)) offered scene recognition functionality. This scene descriptor information was the goal of the Places365 dataset, and was what models were evaluated on while the competition was running. We found the scene attributes to be more beneficial to improve the classification results. However, scene recognition also offered some accuracy increase in our ablation studies, and was therefore included in many of our tested models alongside the scene attributes.

Both scene attributes and the scene descriptor was included as we believed it could give our models more context when classifying the different species. The thought was that some animals may prefer certain geography to others. As an example, goats would be likely to prefer mountainous regions more than deers, if we then had a parameter which told the network that the current image had a high response as a "mountain" image. It would preferentially predict goats, especially in hard to tell cases. Attributes had a similar reasoning, where certain attributes

would be associated more strongly with some species than others.

4.2 Implementation Details

4.2.1 Framework

To create and run the models, we used Python programming language, with PyTorch Paszke et al. (2019) framework for creating, importing, and training models. The models primarily used categorical cross-entropy Zhang and Sabuncu (2018) as the loss function and the Adam optimizer Kingma and Ba (2017).

Training the models was in general done for 50 epochs, this was to assess the capabilities of the different architectures, without requiring an infeasible amount of computational power. While performance could improve beyond the 50 initial epochs, it tended to be near its best performance already. This was enough to assess how well a model may perform at its peak as well, especially in cases where evaluation metrics has significant differences between different models.

The main loss function used was categorical cross entropy, this popular loss function has proved effective in many tasks, and was the most effective in our experiments as well. We also tested other loss functions, such as focal loss Lin et al. (2018) and multi class variations of focal loss.

Optimization was mainly done using the Adam algorithm Kingma and Ba (2017). This optimizer is also highly popular, and was found to be more effective in our case than optimizers such as SGD, ADADELTA Zeiler (2012), and more modern optimizers like LION Chen et al. (2023) which was used in the currently best performing model on "Image Classification on ImageNet" challenge.

We also wished, but did not have the computational capacity, to implement k-fold cross validation. This method increases the overall certainty of the performance of our models. In place of this, we instead used a set seed when testing all models. This makes the dataset split deterministic, as well as the initialization of the weights in a model. We utilized the initial seed 1234 for initialization of all modules that utilized randomness.

4.2.2 Computing power

The networks were mainly created and trained on a Linux computer using an intel-i9-12900KF CPU, 128 Gigabytes of RAM and a RTX3080-Ti GPU. Some networks were too large for practical training on this setup and was either moved to another desktop with a RTX4090 or remote cloud compute power and trained on

a RTX A100 80GB version. The cloud computer was provided by NTNUs central cloud computer solution IDUN[4].

4.2.3 Oversampling and Augmentation

Oversampling was performed using PyTorch Paszke et al. (2019) "WeightedRandomSampler". The augmentation of the oversampled images was performed with an augmentation pipeline using Albumentations python package (Buslaev et al. (2018)). How these augmentation techniques work, is outlined in Section 2.5.3. Each of the augmentation techniques had a certain probability of being applied and a certain limit. We used these limits:

- Horizontal Flipping: $P = 0.5$
- Rotation: $-45° \leq \theta \leq 45°$, $P = 1$
- Color Jitter: Brightness, contrast, hue, and saturation all ± 0.1, $P = 1$
- Droput: size $= 32 \times 32$, holes $= 8$, $P = 1$

4.3 Model Evaluation

This thesis will use several of the evaluation metrics discussed in Section 2.7.1. We will utilize accuracy (abbreviated Acc.), precision (prec.), recall (rec.) F_1 score (F_1), false positive rate (FPR), false negative rate (FNR) and cohen kappa metric (κ). Where applicable, we will use macro averages over micro averages. The exact motivation for this was discussed in Section 2.7.4, but in brief, macro averages lends equal weight to all classes, while macro averages will claim high performance if the majority class is performing well. Performance of minority classes is more important to us in this thesis work than the overall performance of the model.

4.4 Baseline Methods

To properly evaluate if metadata is improving classification problems for WAC, we first need data on performance for networks without metadata applied. This is the purpose of the baseline models. Using the models discussed in Section 2.3, we gave only the image data to the models and evaluated them using the metrics discussed in Section 2.7.

[4] `https://www.hpc.ntnu.no/idun/`

4.5 Ablation Study

Our ablation study focuses on whether metadata can help prediction results at all. This methodology may be somewhat viewed as the opposite of the methodology discussed in Section 4.4. Here we only supply the models with metadata, and do not give any access to the image associated with that metadata.

Ideally, we would investigate which metadata factors impact the prediction accuracy, and which groupings of animals benefit most from which metadata. However, this quickly becomes infeasible. Metadata can be split into: datetime, temperature, position, scene attributes, and scene descriptors. Likewise we have 25 animal classes, or more conservatively 13 classes, we want to investigate. Looking at all pairs, tripples, quads, etc. of animals, together with any permutation of metadata would result in $253,518$ combinations. This is not a feasible number of models to run through for us. Instead we reduced the number of samples to nine (Fox, Deer, Mustelidae, Bird, Lynx, Cat, Sheep, Rodent, Wolf), and combining the relatively short position and temperature information into one combined pos_temp vector, we reduce the number of combinations down to 7529.

To address the issue of imbalanced datasets, we utilized Borderline SMOTE (see Section 2.5.2 for details). We augmented the training data available for the network, but left the validation and testing data untouched.

4.6 Our Models

Our models are based on the ResNet50 architecture, with different modifications that attempt to incorporate metadata into the model. While other models did outperform ResNet50 we selected to use ResNet50 as our backbone for a couple of reasons.

Firstly, the design of the ResNet50 model is in a sense more directional than the Inception v3 block. This makes fusion of metadata with the convolutional blocks easier to do in a sensible way.

Secondly, through our early work on the Snapshot Serengeti dataset, we utilized variations of the ResNet model. This familiarity made it faster to migrate the models over to our main dataset. The minor accuracy improvement Inception v3 models had over ResNet is believed to not have any major effects of the overall conclusions of this paper.

Lastly, we opted for ResNet50 due to its easier comparability to the CBAM model. The supplied resources for CBAM utilized ResNet50 as a backbone, which is then easier to compare against our other architectures.

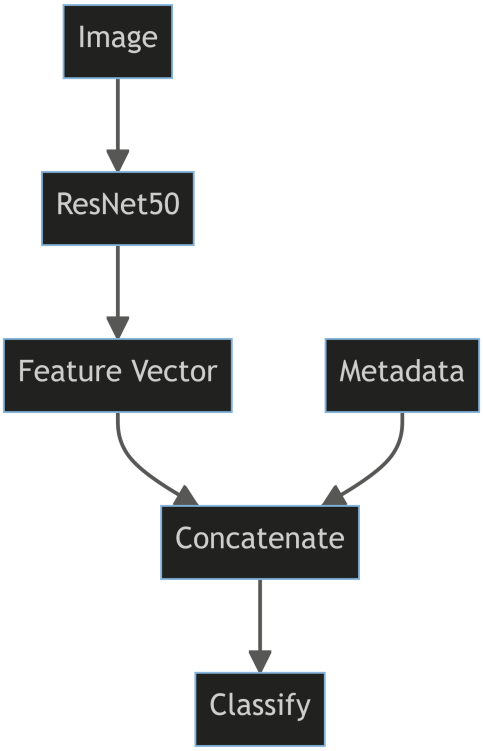


Figure 4.5: *Late Fusion*

4.6.1 Late Fusion Models

The conceptually simplest model we used was the late fusion or decision fusion models. If we have a feature vector $\boldsymbol{v_1} = ResNet50(x)$, which is the final feature vector before classification has occured, and we have $\boldsymbol{v_2} = \mathcal{M}$, which is the metadata. The concatenated vector is $\boldsymbol{v_1} \bigoplus \boldsymbol{v_2} = \boldsymbol{V}$. After concatenation of the vectors, we run the full vector $\boldsymbol{V}$ through three layers of a fully connected network:

$$\hat{y} = g_3(g_2(g_1(\boldsymbol{V})))$$

Here, each of g_1, g_2, and g_3 represent a fully connected linear layer, followed by a ReLU activation function. Conceptually, the model can also be described by Figure 4.5.

4.6.2 Early Fusion Models

The other broad category of fusion is early fusion, or feature fusion, networks. These kinds of network hope to fuse the different data modalities early on in the

process in order to enhance the feature extraction process. Once early fusion is the topic, how to fuse the metadata becomes an open question. Our model applies metadata fusion at three points during the bottleneck block of the Residual Net. This fusion was performed by multiplying the metadata vector by the feature map generated by each convolution block. To ensure that these fit, the metadata was passed through one linear layer with the same output shape as the output of the feature map of the convolution layer. After the linear layer, the metadata was also passed through a Sigmoid function to introduce further non-linearity in our network.

Mathematically, each block of the Residual Net has been modified like this:

$$out = in + g_3(g_2(g_1(c(in) \times f_1(\mathcal{M})) \times f_2(\mathcal{M})) \times f_3(\mathcal{M}))$$

f_1, f_2, and f_3 represent the nonlinear functions that enables the metadata to fit the shape of the feature maps generated by the convolutions. The sign $\times$ here implies a layer wise multiplication between the feature map and the metadata. The different g's represent normalization steps and nonlinear layers the convolution is passed through, and is present in a normal ResNet50 architecture as well. Finally, c represents the first convolution done in each block.

Figure 4.6 gives an overview of how early fusion is performed in our thesis work. Input may be the image, or it may be the output of a previous block.

4.6.3 Modified CBAM Model

This model relies on the CBAM module discussed in Section 2.3.5. However, we included an extra step after channel and block attention was applied to the network. We dub this last component "Metadata attention". Again, looking at the overall structure of the network with our inclusion may be helpful (Figure 4.7).

Matmematically speaking we can define the process as:

$$\begin{aligned} F' &= M_c(F) \otimes F, \\ F'' &= M_s(F') \otimes F', \\ F'' &= M_m(\mathcal{M}) \times F'' \end{aligned} \tag{4.1}$$

Do note that we multiply the metadata vector by the feature maps generated by F'. In line with methodologies used by Li et al. (2020) and Liu (2018). M_m is then defined as:

$$M_m(\mathcal{M}) = \sigma(MLP(\mathcal{M})) \tag{4.2}$$

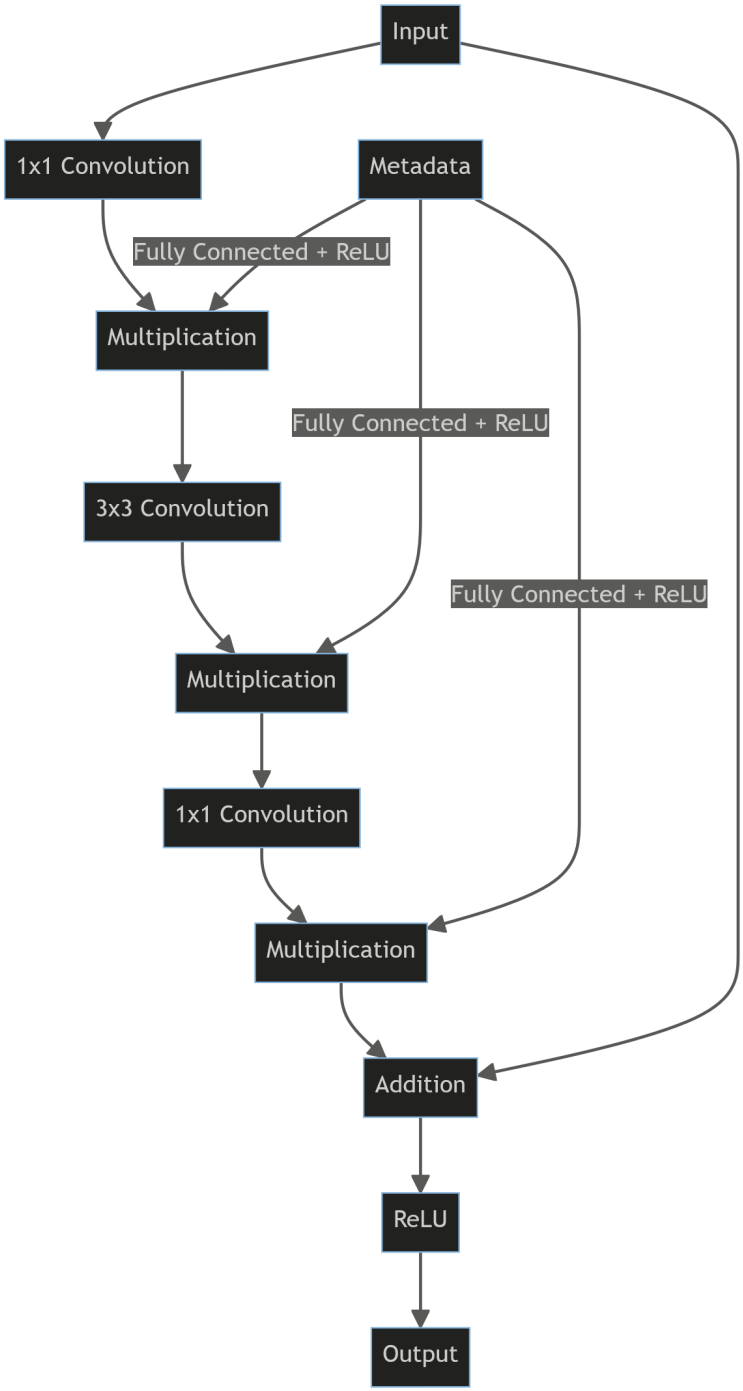


Figure 4.6: *Early Fusion*

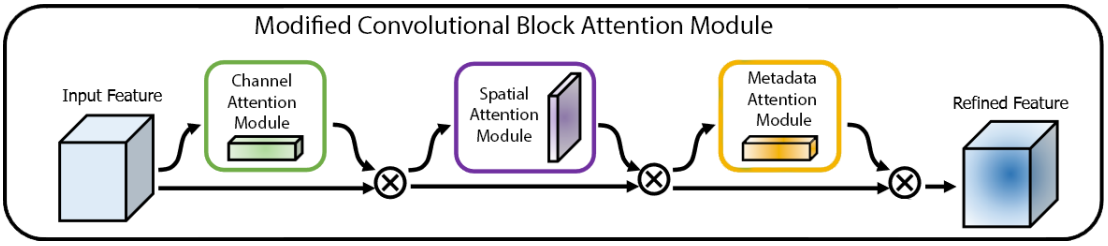


Figure 4.7: *Modified CBAM architecture*

4.6.4 Hierarchical Models

The hierarchical models are based upon findings from the ablation study. The discussion on this can be found in Section 5.2. Since this methodology differs somewhat from the process outlined in Section 4.6, we also included ResNet50 without metadata, to compare the hierarchical models against. As for metadata augmented models, we included both the Late fusion and Early fusion strategies. The goal here is to split the classes into the 13 class list defined in Figure 5.1b. To do this, we first split the dataset into a binary class: "Deer" vs "Not Deer". Deer can then be split into the three subclasses under deer: "Roe Deer", "Deer", and "Capreolinae".

The next steps were influenced by the projections generated by UMAP (see Section 2.6 for further details on UMAP). We want to separate out the classes most easily distinguishable by the metadata. Looking at a reprojection of the metadata, we see mustelidae as the clear separated class (see Figure 4.8).

Once the Mustelidae class is separated, we can rerun the algorithm with the nine remaining classes (Figure 4.9).

We keep removing the most separated class, and rerunning the UMAP projection on the remaining data. And find that we can furhter separate the data into "Fox", "Feline", "Farm Animal", and "Boar" (see Figure 4.10). The remaining classes are not so easily separated by the metadata, as can be seen in Figure 4.11. These hard to separate classes were combined together into one group "Other Animals", that could be classified later. While the "Boar" class was separable using metadata, the number of samples for boars were quite low at only 393. Therefore, boars were also placed into the "Other Animals" group.

The different splits can also be seen in Figure 4.12. Here each label represents an intermediary or final class. The leaf nodes represent the final classes of the hierarchical model.

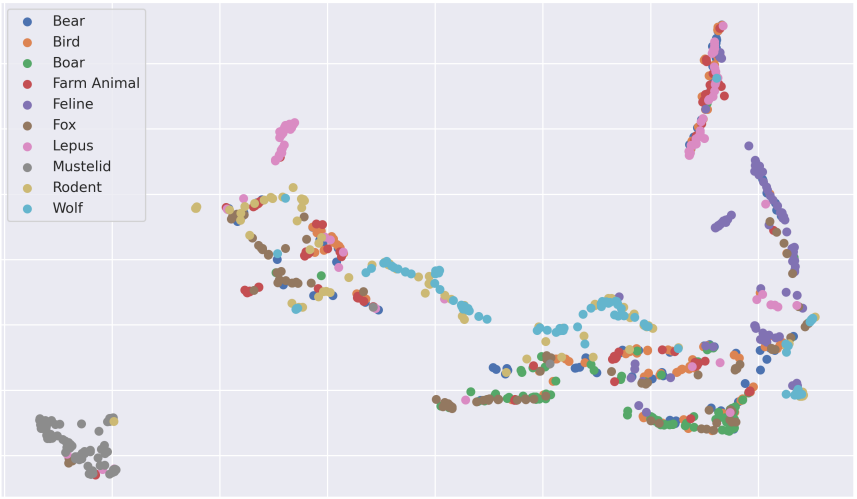


Figure 4.8: *UMAP projection with ten classes*

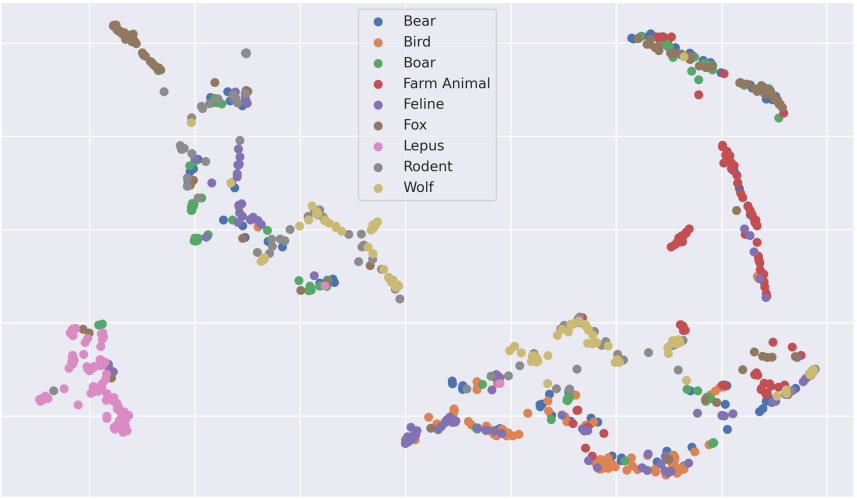
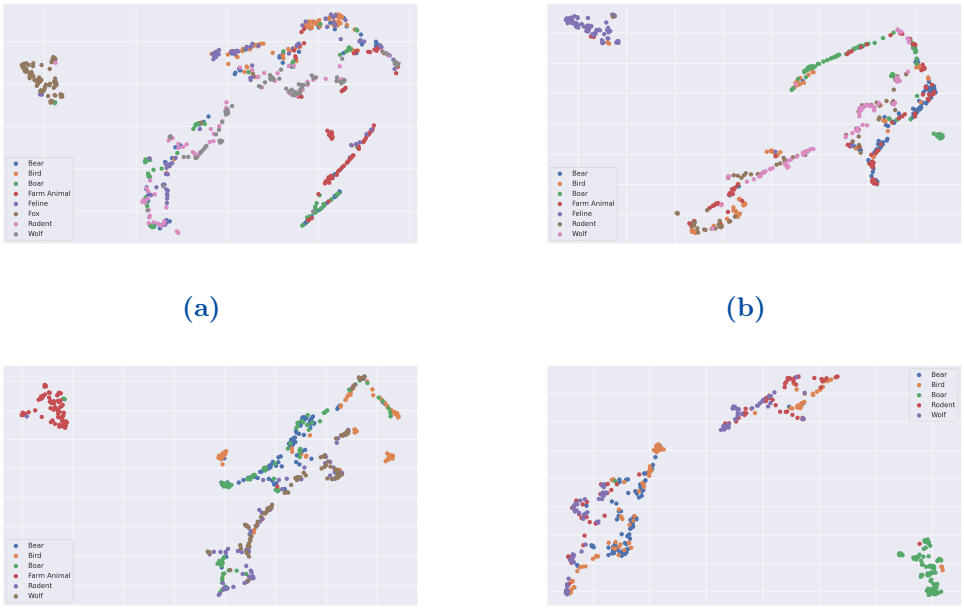


Figure 4.9: *UMAP projection with nine classes*



(a) (b)

(c) (d)

Figure 4.10: *UMAP Projections demonstrating a separation between classes*

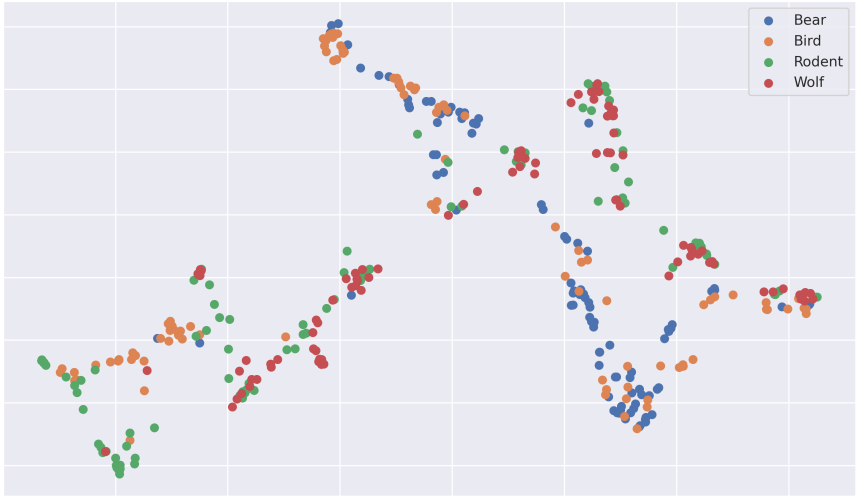


Figure 4.11: *UMAP no longer cleanly separates the classes*

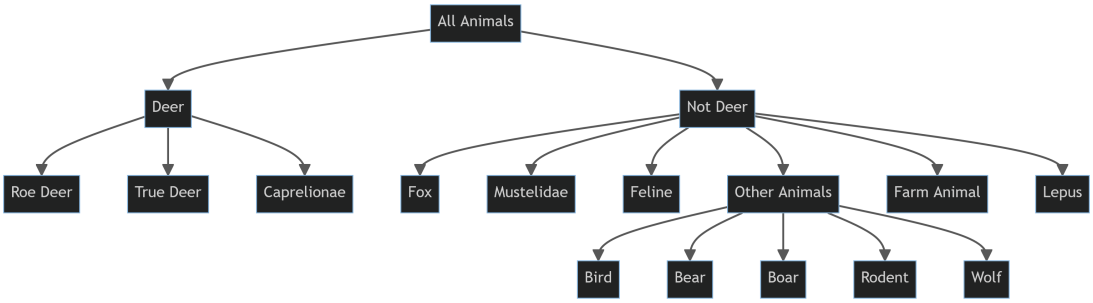


Figure 4.12: *Hierarchical Model split*

4.7 Challenges

4.7.1 Data Challenges

There were three main issues with the data: collection, completeness, and validation

Collection

The NINA Viltkamera dataset had no formal way to access it. This forced us to collect it via web scrapers. This was done with the knowledge of NINA. However, to limit the load on their servers, we throttled the download rate of the dataset to 5 images per second. Collecting all 170 thousand images at a rate of 5 images per second, lead to a lot of waiting for the dataset to become complete.

Access to the dataset was supposed to be in collaboration with NINA, and they were working on providing us a central location for downloading a large amount of their available data. Creating this proved to be more challenging than expected, and lead to the final data being available later than expected. To alleviate this

issue, we used the SS dataset (Swanson et al. (2015)) in the interim to test models and methodologies.

Completeness

As previously discussed in Section 4.1.5, some images lacked complete metadata information. Especially temperature data was lacking from several samples. To combat this issue, we first tried to fill in the samples by using available metrological data, along with location and date. This left us with a few missing values that we had to handle. We considered setting them to the average temperature, or set the value to zero. However, both were problematic as zero degrees is not an uncommon temperature in Norway, and the average temperature still means something. We ultimately decided to use a 2-dimensional vector for the temperature values. Where the first value told us whether the temperature was valid via a binary digit, and the second value was the actual temperature. In the case where we don't know the temperature, we simply set the second value to zero.

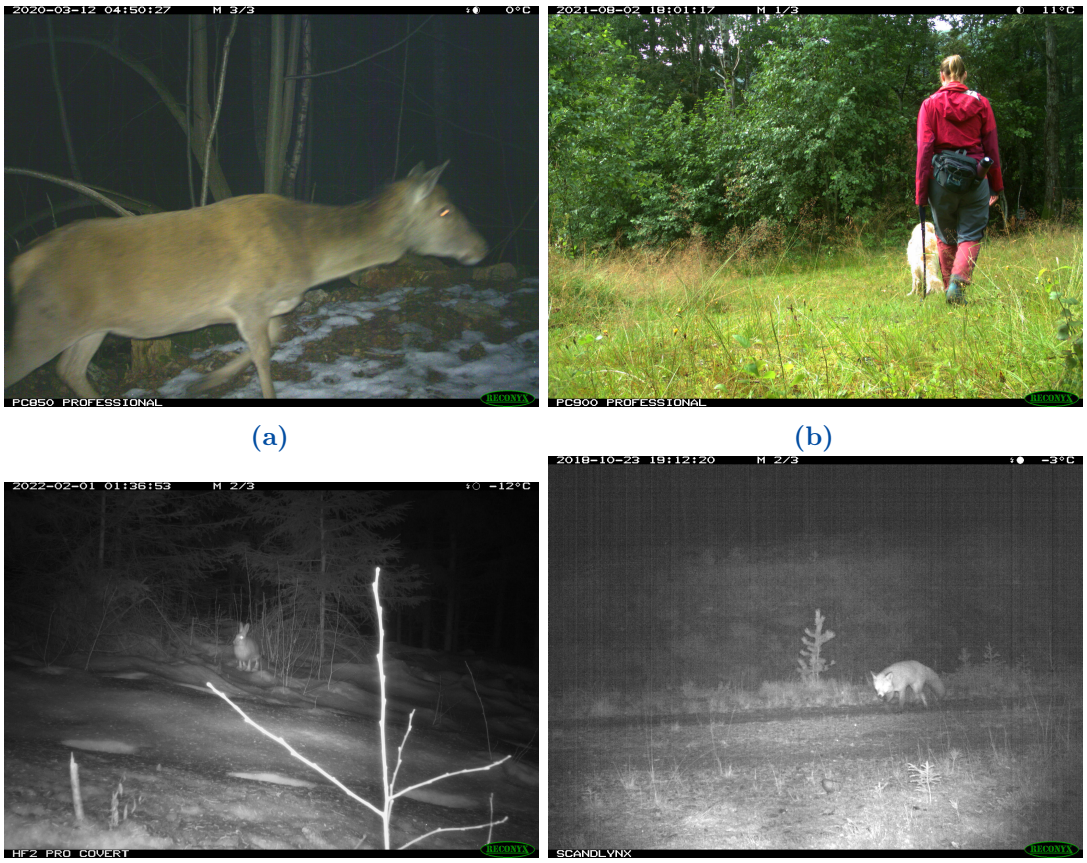
Validation

The issue with using a less common dataset is the lack of validation on said dataset. Several samples with one given class were in fact a different class (see Figure 4.13). Unfortunately, due to the sheer number of samples, combined with the lack of relevant expertise from the authors of the paper, reclassifying the animals is infeasible. Luckily, the vast majority of labels are correct, with only around $0.5\% - 1\%$ of labels being wrong.

4.7.2 Computational Challenges

Deep learning is a computationally heavy task. With millions of tunable parameters, being updated thousands of times, the total computation done is mind boggling. This thesis would also benefit from more computational power. Extra power would let us more quickly assess good models and bad models. Which may have enabled the creation of better architectures that could outperform the baseline models.

However, it may not always be beneficial with more power. We did a quick sidetrack to get a rough estimate for how much CO_2 was emitted during this thesis work. This small sidenote can be found in Appendix D.



(a) (b)

(c) (d)

Figure 4.13: *Animal misclassifications. All are labeled as "Sheep"*

4.7.3 Methodological Challenges

There were several methodogial issues that ran throughout this thesis work. The first one we will discuss is the lack of expertise from the thesis writer. This issue comes mainly in the form of how to combine, group, and prioritize the animals in the NINA dataset. Fairly naive, and suboptimal methods were used at first to separate the original 65 classes. The first method relied on simple pruning of the minority classes with too few samples to be realistically classified. This threshold was set to roughly 100 samples, meaning any of the samples outlined in Table A.1 with fewer than 100 datapoints would be removed. This is not ideal, as ecologists often care more about the species with fewer samples related to them. These species are most likely to be endangered, and therefore require more monitoring. Later methods instead combined these animals based on common taxa, which keeps some of the data intact (removing the original label, replacing it with a more generic

one).

Furthermore, some of the models, especially the older ones, were created using a sub-optimal pipeline. This meant that instead of defining new architectures for each model, a previous model was modified in response to issues the researchers discovered with the previous model run. This made reproducibility harder to achieve, as it was not always possible to recover the original model based on the notes made by the author. The results could still be somewhat analyzed, since a confusion matrix was stored for every model run, but it was suboptimal. Later models were always defined as their own separate class, regardless of how minor changes were from previous models. And the last step was to save the entire model architecture as a binary file along with the final weights for each model, making evaluation easier.

5 Results and Discussion

This chapter is dedicated to displaying results from data acquisition, as well as model performances after training the various networks.

5.1 Data Acquisition

5.1.1 Snapshot Serengeti

In total, seasons one through six were collected. As well as the image labels and bounding boxes where those existed. We elect not to discuss this data in much detail, as it was primarily used for early experimentation and learning. However, some discussion on our findings can be valuable.

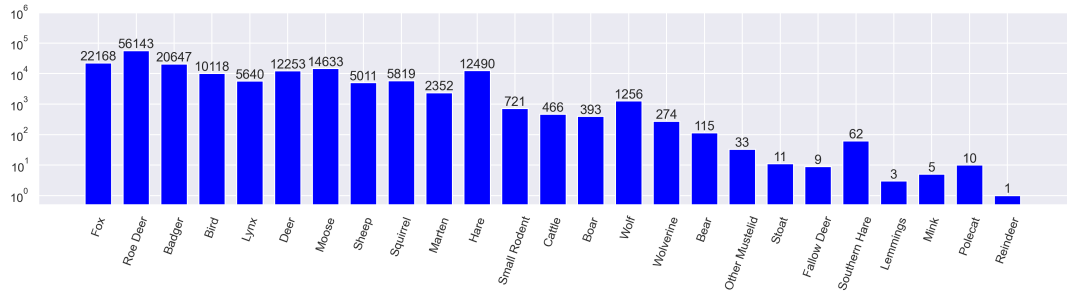
We could then fetch the image during training by reading the image path from the metadata JSON object, and fetching that specific image. This strategy revealed one issue in our methodology: fetching images from a Hard Disk Drive is terribly slow. The GPU utilization was around 20% while the disk usage was 100% constantly. A couple of solutions were considered here. Hadoop Distrubted File System[1], Lustre[2], and WekaFS[3]. The problem with all of these file systems is their distributed nature. These file systems are more aimed at distributed computing problems, where we use several computing and data nodes which work together to solve a task. Since we did not utilize cloud computers and multiple nodes for the majority of the project, it would only increase the complexity of the solution while providing only minor benefits. Instead, we focused on storing as much of the data as possible on faster Non-Volatile Memory Express (NVMe) Solid State Drives (SSD). This solved our problem, letting us fully utilize the GPU for network training.

[1] `https://hadoop.apache.org/`
[2] `https://www.lustre.org/`
[3] `https://www.weka.io/resources/datasheet/wekafs-the-weka-file-system/`

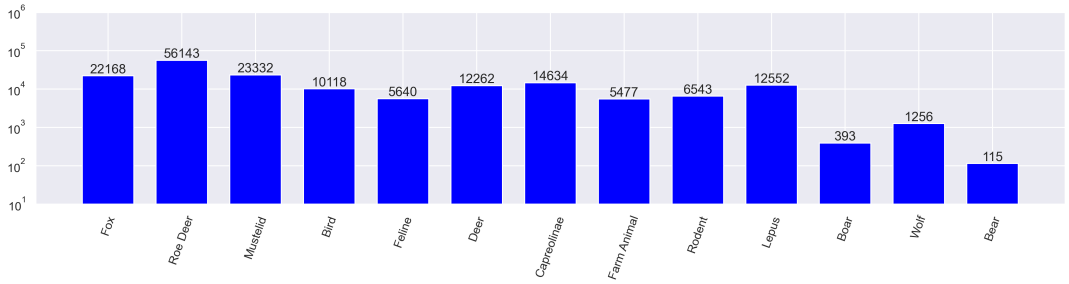
5.1.2 Nina Viltkamera

Description and Statistics

The NINA Viltkamera dataset had a total of 100 classes that images could be classified as. However, not all 100 classes had any samples in them. Only 65 classes in the downloaded dataset had one or more samples associated with them. The class labels and count can be seen in Table A.1. Creating a good classifier for all 65 classes would be quite challenging, especially considering the low number of samples for some of the classes. We therefore combined them into super-classes. A mild combination can be seen in Figure 5.1a. Note that due to the significant variance in the number of samples per class, we used a log scale in the figure. We have included the total number of samples per class above the bar for easier reading. The most significant combination here is combining all birds into one super-class "Bird". This 25-class dataset is still quite imbalanced, and while the larger classes like "Roe Deer" will fare well, we worry some of the smaller classes will be ignored. Therefore, we created a more aggressive grouping of the classes, which can be seen in Figure 5.1b. This 13-class dataset is what we will use as our initial dataset, which can be expanded later to include some or all of the more specific labels from the 25-class distribution.



(a) *Mild grouping of animals*



(b) *Aggressive grouping of animals*

Figure 5.1: *Data distribution*

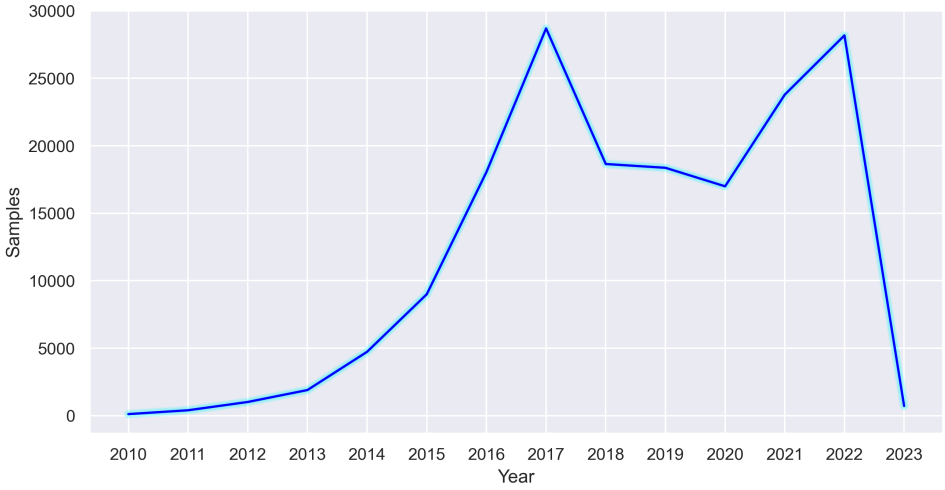


Figure 5.2: *Distribution of samples by year.*

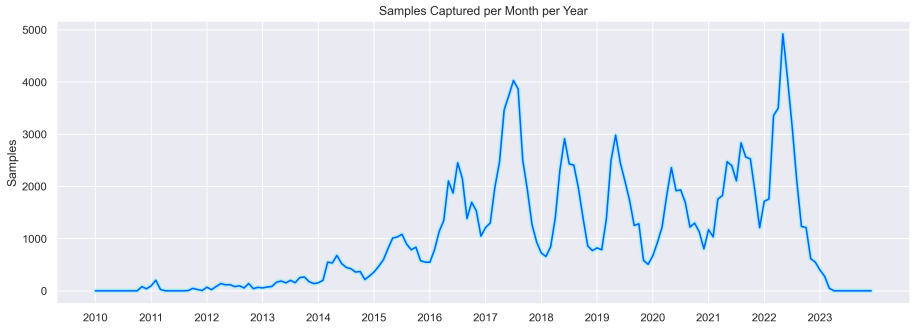


Figure 5.3: *Distribution of samples by year and month.*

Over on the temporal side we can observe a couple of things about the data. Most of the data was collected after 2016 (Figure 5.2), with the largest year of collection being in 2017. 2023 is an outlier here since the cutoff date for data collection was March 11th, but the project is still ongoing, and more data is expected to be added in 2023 and forward. We can also observe that data is collected over the years in a cyclical pattern (Figure 5.3), with most of the data being collected in the early summer of the year. This is clearer if we look at the aggregate days of the year for data collection, which can be seen in Figure 5.4.

Finally, we can observe some samples from the different classes. This can give us an idea of the challenges the network needs to overcome in order to classify the different classes. Figure 5.5a displays a deer somewhat camuflaged due to its hairs

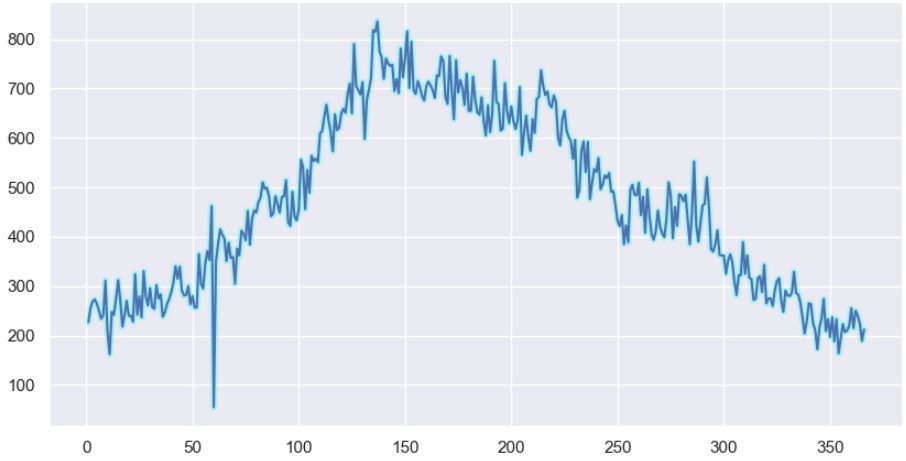


Figure 5.4: *Seasonal distribution of data. Downward spike on day 60 of the year is due to leap years.*

while Figure 5.5b demonstrates the issues of the blitz function during nighttime. We may also contend with partial images like Figure 5.5c and 5.5d. While these images are challenging, they are not the hardest images available. Some images have only slight bristles hair visible (Figure 5.6a), or are so blurry it is hard to make out anything at all (Figure 5.6b).

5.1.3 Metadata

The missing temperature data was successfully aquired as described in Section 4.1.5. The original dataset of 170 thousand samples, contained 90 thousand samples where temperature data was missing. After we performed external temperature acquisition with the Frost API, we were left with only 16 thousand samples where temperature data were still missing. While not ideal to have a missing datapoint for roughly 9% of our data, it is much preferable over the initial 53% of temperature data that was lacking before fetching temperatures from the Frost API.

5.2 Ablation Study

The purpose of the ablation study was to find whether metadata could tell the difference between the animals at all, this topic was more closely discussed in Section 4.5.

(a) (b)

(c) (d)

Figure 5.5: *Image variety and challenges*

(a) (b)

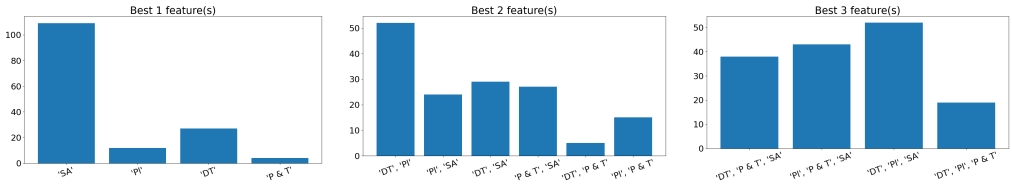
Figure 5.6: *Challenging images*

We've decided to use an ID for each species instead of the said species' name. The corresponding ID to species is 0: 'Fox', 1: 'Deer', 2: 'Mustelidae', 3: 'Bird', 4: 'Lynx', 5: 'Cat', 6: 'Sheep', 7: 'Squirrel', 8: 'Rabbit', 9: 'Rodent', 10: 'Cattle', 11: 'Boar', 12: 'Wolf', and 13: 'Bear'. We can look at the best prediction results for m classes when using n metadata types. The metadata was closer discussed in Section 4.1.5. In Table 5.1, we see that the "Scene attributes" information yields the best single feature to include in the prediction. We also see as we increase the number of features included performance also increases.

However, the average performance of the different features is less clear-cut. We can quantify this relation better by looking at the "winner" when comparing the performance of n predictors against each other to predict between m classes. By finding and counting the best predictor(s) for all combinations of animals, we get Figure 5.7. To save space, we used abbreviated versions of the feature names, 'SA' equates to scene attributes, 'Pl' is short for "Places" which are the Scene descriptors, 'DT' is the datetime vector, and 'P & T' is the position and temperature information. We see that "Scene attributes" is the clear best single predictor. However, it is not among the pair of best predictors, being beaten out by the combination of "Datetime" and "Places". Its worth noting that this method of counting the winner does not take into account how much better one predictor performed than another. We do not know whether "Scene Features" dominated the competition as the singular feature or if other features were close seconds to the best performance of "Scene Features". However, we can conclude

Table 5.1: *Metadata Predictors Scores*

Classes	Features used	Acc	κ
4, 6	Scene attributes	0.948	0.894
6, 12	Position and temperature, Scene attributes	0.982	0.945
4, 6	Places, Position and temperature, Scene attributes	0.967	0.932
6, 12	Datetime, Places, Position and temperature, Scene attributes	0.989	0.964
3, 4, 6	Scene attributes	0.87	0.779
3, 4, 6	Position and temperature, Scene attributes	0.869	0.782
3, 4, 6	Datetime, Places, Scene attributes	0.866	0.775
3, 4, 6	Datetime, Places, Position and temperature, Scene attributes	0.878	0.796
2, 3, 4, 6	Scene attributes	0.696	0.552
3, 4, 6, 12	Position and temperature, Scene attributes	0.731	0.603
3, 4, 6, 12	Datetime, Position and temperature, Scene attributes	0.729	0.614
3, 4, 6, 12	Datetime, Places, Position and temperature, Scene attributes	0.746	0.63



(a) **(b)** **(c)**

Figure 5.7: *The best n features to use to distinguish a set of m animals*

that accuracy, in general, improves when more features are included. Meaning all the metadata contributes something valuable to the prediction of the animal feature. Remember that these predictions of animal classes are purely based on the metadata information, no image of the animal is given to the model, yet it can quite confidently predict between two classes.

Lastly, we can note that as we introduce more classes, the performance of prediction goes down. We can see this trend more clearly in Figure 5.8. This is an understandable trend, as guessing between two classes is easier than guessing between ten. However, the kappa score takes account for the random guess factor, but this also goes down. This suggests that as we increase the number of classes, the differentiating power of metadata is reduced. This suggests that to get the

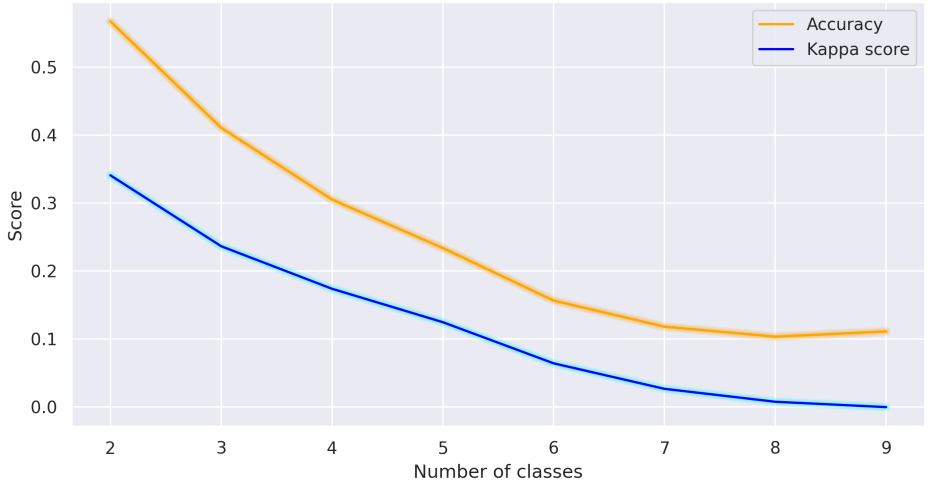


Figure 5.8: *Prediction score versus number of classes to distinguish*

maximum effect of differentiation from metadata, we should focus on cases where we have as few classes as possible. And we need to ensure that those classes are distinct when looking at the metadata.

5.3 Complete models

5.3.1 Results

We will first first lay out the results of most of our models, before discussing those results in the next sections. The results for all full models can be seen in Table 5.2. The only models not included in the results are the hierarchical models. The hierarcical model will be discussed later in Section 5.4.

5.3.2 Discussion

Baseline models

The result of unmodified or "Baseline" models are the first few entries in Table 5.2. These results reflect the performance of models that only have minor to no modifications from the proposed methods outlined in the original papers the models were a part of. These papers and their methodology can be seen in Section 2.3. The purpose of these models is to compare our models against something known.

Table 5.2: *Baseline model results*

Model	Acc.	Prec.	Rec.	F_1	FPR	FNR	κ
ResNet18	0.966	0.953	0.964	0.958	0.003	0.036	0.959
ResNet50	0.983	0.974	0.978	0.976	0.001	0.022	0.98
AlexNet	0.888	0.841	0.909	0.872	0.01	0.091	0.867
EfficientNet B3	0.982	0.98	0.979	0.979	0.002	0.021	0.978
Inception v3	0.984	0.981	0.979	0.980	0.001	0.021	0.980
CBAM	0.805	0.688	0.789	0.720	0.017	0.211	0.768
Late fusion	0.987	0.986	0.98	0.983	0.001	0.02	0.984
Early fusion	0.989	0.984	0.986	0.984	0.001	0.014	0.987
MCBAM	0.795	0.703	0.783	0.735	0.018	0.217	0.758

We can infer the efficacy of metadata augmentation by seeing how well metadata augmented models function, compared to a variety of models not augmented by metadata.

We can observe that the general accuracy of models is already quite good, making any improvements relatively small percentage wise. Of the baseline models tested, the Inception v3 architecture has minor performance advantage over the other models. With ResNet50 coming at a close second. While only EfficientNet B3 is shown here, we did perform testing of all EfficientNet iterations and used only the best performer in our further testing.

Some of our early experiments tested if the performance would be improved if we had smaller datasets. Our theory was that the extra data may help a network learn faster, conversly it may also learn slower, as more parameters needed tuning. We artificially removed 70% of the original dataset for training, and then compared performance of metadata augmented versus non-augmented models. We found no performance increase in our experiments, suggesting that the extra data does not make up for the extra complexity introduced into the models by including metadata.

Our Models

The results of our models can also be seen at the end of Table 5.2. We see that two of our three models perform at the same or better than our baseline models. MCBAM is the only model that does not seem to benefit from the inclusion of metadata, dropping by a whole percentage in prediction accuracy. MCBAM does have a higher overall precision than the baseline CBAM model. However, the gain in precision does not make up for the drop in the recall. This may still be desireable, as a higher precision suggests that we are less likely to predict the majority class.

We can evaluate this better by comparing CBAM and MCBAMS accuracy per class.

Table 5.3: *Comparison of the accuracy per class of the CBAM and MCBAM models with counts for each species*

Class	CBAM	MCBAM	Count
Fox	0.746	0.699	22168
Roe Deer	0.829	0.792	56143
Mustelid	0.784	0.804	23332
Bird	0.823	0.777	10118
Feline	0.775	0.716	5640
Deer	0.793	0.903	12262
Capreolinae	0.818	0.877	14634
Farm Animal	0.897	0.956	5477
Rodent	0.679	0.719	6543
Lepus	0.855	0.798	12552
Boar	0.846	0.786	393
Wolf	0.87	0.69	1256
Bear	0.545	0.667	115

We see here that CBAM is generally better at the majority class classifications. While MCBAM performs slightly better on the minority class samples. The tradeoff here has to be determined on a case by case basis, based on the needs of the ones using the models. It is still promising to see that the inclusion of extra data from metadata, does not mean that a model will be more likely to predict the majority class.

Our late fusion model does outperform the best baseline models by a small margin. However, we are more interested in the early fusion model, as this one performs slightly better again. This observation of early fusion outperforming late fusion is in line with other papers that discuss the matter (Arevalo et al. (2017), Liu (2018), Li et al. (2020)).

We can once again compare the performance on a class by class basis. Table 5.4 shows us that only a few classes are now better classified by the best baseline model. This is especially promising, as we only tested a couple of architectures for metadata augmented models. We are certain other models will outperform the models tested here, one may even consider constructing architectures from the ground up, with metadata augmented feature extraction incorporated in from the start.

Table 5.4: *Comparison of the accuracy per class of the Inception v3 and Early Fusion models with counts for each species*

Class	Inception v3	Early Fusion	Count
Fox	0.985	0.99	22168
Roe Deer	0.981	0.986	56143
Mustelid	0.982	0.99	23332
Bird	0.981	0.987	10118
Feline	0.991	0.989	5640
Deer	0.988	0.997	12262
Capreolinae	0.994	0.994	14634
Farm Animal	0.993	0.995	5477
Rodent	0.968	0.988	6543
Lepus	0.986	0.99	12552
Boar	0.964	1.0	393
Wolf	0.992	0.992	1256
Bear	0.917	0.917	115

5.4 Hierarchical Models

The results of running the hierarchical models were not promising. The partial results can be seen in Table 5.5. Only overall accuracy and kappa score has been included, as some classes failed to get a single sample, making metrics such as precision ill defined. The issues with the hierarchical model could already be seen at the first step of training, where deers were separated from non-deers. The accuracy at this step was around 98% for the validation set. Meaning, regardless of how well the model performed on the sub-classes, it was bound to have an accuracy of 98% or less.

Table 5.5: *Hierarchical Classification Results*

Model	Acc.	κ
ResNet50	0.412	0.292
Late Fusion	0.412	0.294
Early Fusion	0.410	0.291

The results of the hierarchical models indicates that one should be careful when fusing metadata and image data together. Not every method available will result in improved performance. We do not have enough data to say for certain that hierarchical models are not the way forward, but we can conclusively say that our methodology did not work as intended.

5.5 Typical Misclassification

This section aims to demonstrate some typical misclassifications done by a well-trained network. These networks range in the 89-98% accuracy threshold, and generally performs well. Some misclassifications are hard to say why are miss-classified. These images are in general clear, and should not be difficult to tell which animals are which.

(a) *Classified as Sheep*

(b) *Classified as Bird*

(c) *Classified as Boar*

(d) *Classified as Bear*

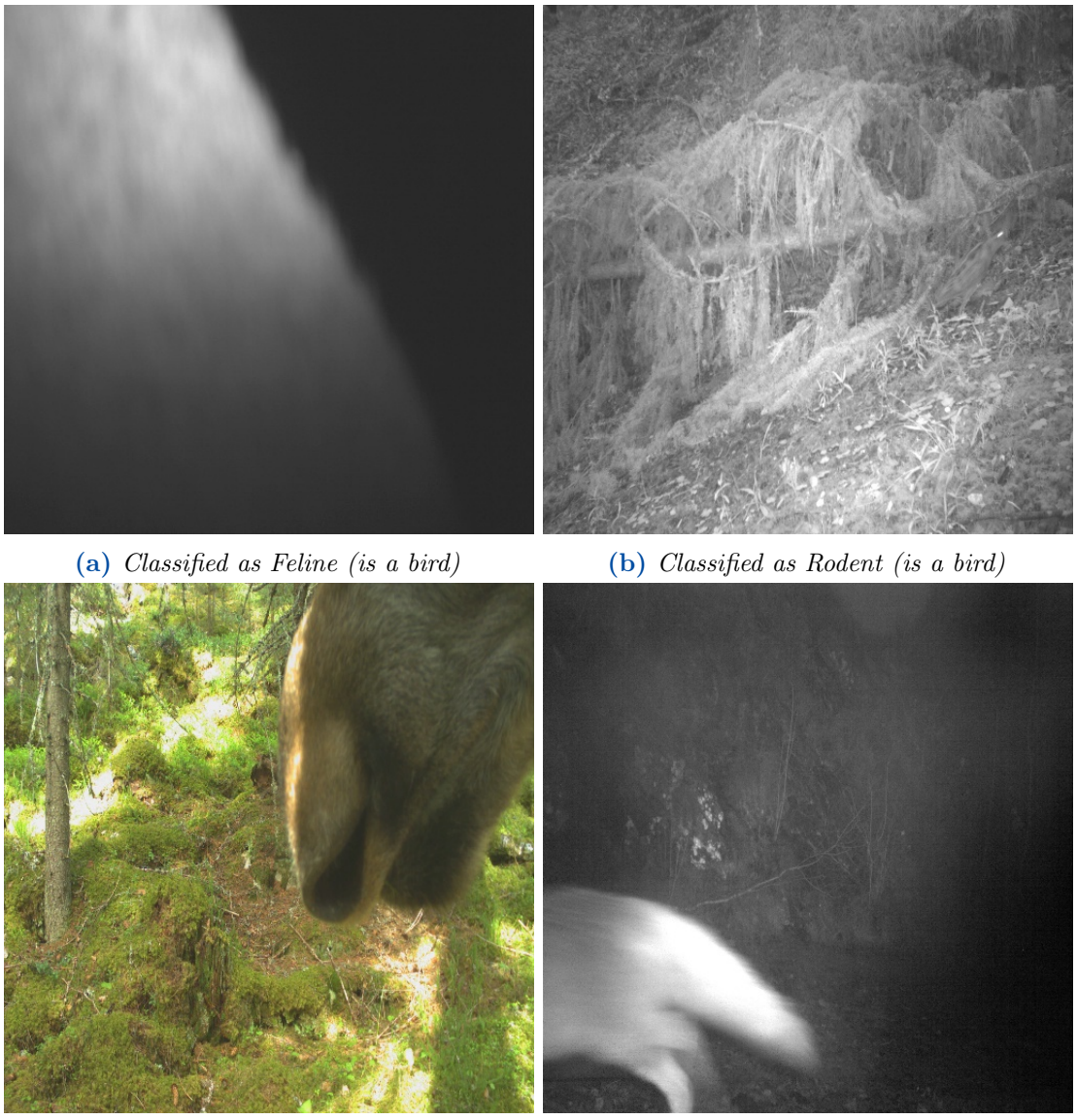
Figure 5.9: *Example of poorly classified images*

We used an earlier ResNet18 based late fusion model to find some examples of wrongly predicted images. While the model is a bit older than the final versions used in the thesis, it still achieved an accuracy of 96%, and should be representative of the sorts of misclassified samples we get in the best models as well. The examples in Figure 5.9 are images that should be properly classified. However, the easily classifiable images, are for some reason still misclassified. This is one of the challenges with neural networks. You may not always know why an image is misclassified. Figure 5.9d is maybe the only image that is hard to tell. The glare of the image significantly degrades the quality of the image, causing classification to be harder. We also have other images that are significantly harder to classify. Images in Figure 5.10 are some examples of where the model predicted wrong label, but we can give it some leeway, as the images are already really hard to tell what animal it actually is. Either the image is partial, and has a lot of missing context. Images can also be blurry, making classification more challenging as well.

One interesting image here is Figure 5.10b. This image, with a true label "Bird" is classified as "Rodent". The model may have learned that these seemingly empty images are more likely to contain small animals that are typically associated with the "Rodent" class. We can investigate if this is the case by looking at the networks reaction when feeding empty images into it. To get some empty images, we utilized the Caltech Camera Trap project (see Section 4.1.1 for further details on this dataset).

Using the same ResNet 18 model that gave the image results shown in Figure 5.10b, we find that "Bird" is the most popular class to predict when an empty image is presented. This is followed by the class "Rodent". The total distribution of predictions can be seen in Figure 5.11. Since "Bird" is the most popular classification result when presented with empty images, one may wonder why a "Rodent" was predicted. This phenomena is here caused by a sampling bias. The images above are only samples taken when the predicted class is wrong.

The samples displayed in Figure 5.9 and Figure 5.10 are displayed because of automated flagging when a sample was wrongly predicted. One of the flagged images was a seemingly empty image. Our model is most likely to predict "Bird" when given a seemingly empty image. However, if our model predicted "Bird" in this case, we would not flag it as wrongly predicted. We unfortunately have no real data on which images in the NINA Viltkamera dataset that are empty, as no images has the label empty. But, we can probably assume that a decent amount of images labeled as "Bird" and "Rodent" may in fact be empty, or at least seemingly empty to the camera. We can summarize that the network has learned that an empty image means there is a small animal present in the image. Since the network cannot locate that animal, it just guesses based on the frequency of small animals.



(a) *Classified as Feline (is a bird)*

(b) *Classified as Rodent (is a bird)*

(c) *Classified as Bear (is a Caprelionae/Moose)*

(d) *Classified as Bird (is a fox)*

Figure 5.10: *Example of hard to tell misclassified images*

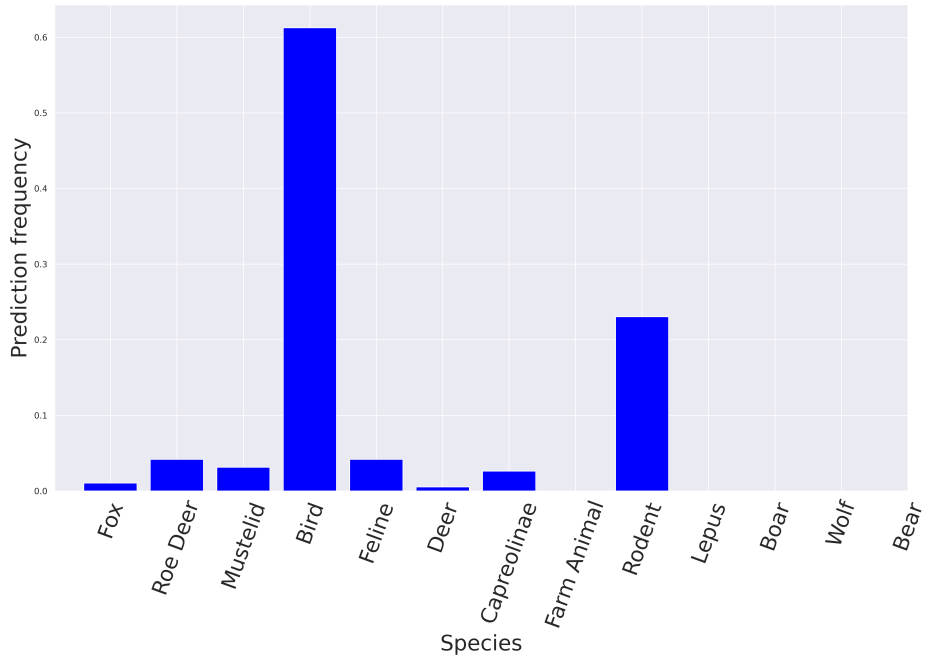


Figure 5.11: *ResNet18 prediction distribution on empty images*

6 Conclusion and Further Work

6.1 Conclusion

This thesis work demonstrated that the use of metadata can be used as a classifier of wild animals, especially when few different classes are involved. The strength of the classifier goes down as more animal species are included. We also demonstrated that more metadata will increase performance. However, the increase in metadata features has diminishing returns. The strongest metadata feature found in our work was Scene Attributes, which can be automatically extracted from any image.

We also demonstrated that this metadata can give a increase in performance to deep learning models using both image data and metadata. This connection was clearest in our early fusion model. However, our experiments are not extensive enough to properly validate if this result extends to other Wild Animal Classification projects outside the Norwegian setting.

In conclusion, while the incorporation of metadata has been shown to potentially enhance the performance of deep learning models, our study suggests that this is not universally true for all models and caution must be exercised to prevent overfitting. The increased number of parameters introduced by metadata necessitates a thorough evaluation and comparison of models with and without metadata. While our study is one of the first in the field of Wild Animal Classification to explore the use of metadata, it is crucial to recognize that there is no established precedent to guide our expectations. Thus, while the successful implementation of data fusion in other deep learning domains provides a promising outlook, it would be premature to assume its universal applicability in the field of Wild Animal Classification without further extensive research and experiments.

6.2 Further Work

We recommend further investigations into the use of metadata enhanced Wild Animal Classification. Especially the usage of automatic scene attribute and

descriptor detectors such as the ones proposed by Zhou et al. (2017). These automatic models can be implemented into Image Classification topics without the need for manual annotation by a human expert. Providing extra value to a dataset essentially for free.

Furthermore, we advice camera trap projects to as much as possible provide metadata for captured data. We utilized temperature, location, and time from the captured samples, but encountered issues collecting especially temperature data from the original set. Information such as air pressure and humidity could also be relevant factors we did not have the options to explore. Easy access to these kinds of data makes creation of models relying on metadata easier as well.

A Table of classes in NINA dataset

Table A.1: *All species in NINA dataset*

Animal	Count
Fox	22168
Roe Deer	56143
Badger	20647
Bird	6882
Lynx	4002
Deer	12253
Moose	14633
Cat	1638
Sheep	5011
Squirrel	5819
Marten	2352
Woodcock	287
Hare	12490
Grouse	429
Blackbird	511
Jay	59
Great Tit	266
Small Rodent	721
Cattle	466
Wild Boar	393
Bird of Prey	6
Wood Pigeon	286
Nuthatch	4
Tit sp.	61
Song Thrush	124
Thrush sp.	124

Continued on next page

Table A.1 – continued from previous page

Animal	**Count**
Capercaillie	448
Wolf	1256
Wolverine	274
Bigfoot	0
Bear	115
Black Grouse	42
Tawny Owl	8
Blue Tit	32
Other Mustelid	33
Stoat	11
Buzzard	7
Wood Grouse	60
Crow	12
Magpie	170
Canada Goose	2
Crane	32
Fieldfare	49
Pied Flycatcher	12
Fallow Deer	9
Black Woodpecker	20
Great Spotted Woodpecker	10
Redwing	52
Bullfinch	14
Southern Hare	62
Lemmings	3
Raven	54
Grey Heron	2
Peregrine Falcon	1
Mink	5
Polecat	10
Chaffinch	15
Mistle Thrush	19
Robin	9
Green Woodpecker	1
Great Grey Owl	1
Reindeer	1

Continued on next page

Table A.1 – continued from previous page

Animal	**Count**
Nutcracker	3
Siberian Jay	2
Greenfinch	1
Spotted Flycatcher	1

B | Web Scraper code

```
function sleep(ms) {
  return new Promise(resolve => setTimeout(resolve, ms));
}

async function downloadData() {
  // Get all pins
  let pins = document
              .querySelector('.leaflet-marker-pane')
              .childNodes;

  for (let i = 0; i < pins.length; i++) {
    pins[i].click();
    // 1 second delay to let vm object load
    await sleep(1000);

    // fetch the location from dynamic HTML tag
    let h2Content = document
                  .querySelector('#dialogheader')
                  .textContent;
    const number = h2Content.match(/Lokalitet:\s(\d+)/)[1];
    const lokalitetId = parseInt(number);
    let lokaliteter = (vm.lokaliteter())

    // find the correspondign long/lati from location id
    const obj = lokaliteter
        .find(o => o.LokalitetID === lokalitetId);
    const latitude = obj.Latitude;
    const longitude = obj.Longitude;

    // add longitude/latitude to each object of vm.media
    let json = vm.media.map(obj => (
```

```
                    {...obj,
                        latitude: latitude,
                        longitude: longitude
                    }));

        // Create a blob from the JSON object
        let blob = new Blob(
            [JSON.stringify(json)],
            { type: 'application/json' }
            );

        // Create and click a download link for the file
        let downloadLink = document.createElement('a');
        downloadLink.href = URL.createObjectURL(blob);
        downloadLink.download = 'data' + i + '.json';
        downloadLink.click();
    }
}

downloadData();
```

C | Places Attributes and scenes

Scene attributes

boating	driving	biking
transporting	sunbathing	touring
hiking	climbing	camping
reading	studying	training
research	diving	swimming
bathing	eating	cleaning
socializing	congregating	waiting in line
competing	sports	exercise
playing	gaming	spectating
farming	constructing	shopping
medical activity	working	using tools
digging	conducting business	praying
fencing	railing	wire
railroad	trees	grass
vegetation	shrubbery	foliage
leaves	flowers	asphalt
pavement	shingles	carpet
brick	tiles	concrete
metal	paper	wood
vinyl	plastic	cloth
sand	rock	dirt
marble	glass	surf
ocean	running water	still water
ice	snow	clouds
smoke	fire	natural light
sunny	indoor lighting	aged
glossy	matte	sterile
moist	dry	dirty
rusty	warm	cold
natural	man-made	open area
semi-enclosed area	enclosed area	far-away horizon
no horizon	rugged scene	vertical components
horizontal components	symmetrical	cluttered space
scary	soothing	stressful

Scene Descriptors

airfield	airplane_cabin
airport_terminal	alcove
alley	amphitheater
amusement_arcade	amusement_park
apartment_building/outdoor	aquarium
aqueduct	arcade
arch	archaelogical_excavation
archive	arena/hockey
arena/performance	arena/rodeo
army_base	art_gallery
art_school	art_studio
artists_loft	assembly_line
athletic_field/outdoor	atrium/public
attic	auditorium
auto_factory	auto_showroom
badlands	bakery/shop
balcony/exterior	balcony/interior
ball_pit	ballroom
bamboo_forest	bank_vault
banquet_hall	bar
barn	barndoor
baseball_field	basement
basketball_court/indoor	bathroom
bazaar/indoor	bazaar/outdoor
beach	beach_house
beauty_salon	bedchamber
bedroom	beer_garden
beer_hall	berth
biology_laboratory	boardwalk
boat_deck	boathouse
bookstore	booth/indoor
botanical_garden	bow_window/indoor
bowling_alley	boxing_ring
bridge	building_facade
bullring	burial_chamber
bus_interior	bus_station/indoor

butchers_shop	butte
cabin/outdoor	cafeteria
campsite	campus
canal/natural	canal/urban
candy_store	canyon
car_interior	carrousel
castle	catacomb
cemetery	chalet
chemistry_lab	childs_room
church/indoor	church/outdoor
classroom	clean_room
cliff	closet
clothing_store	coast
cockpit	coffee_shop
computer_room	conference_center
conference_room	construction_site
corn_field	corral
corridor	cottage
courthouse	courtyard
creek	crevasse
crosswalk	dam
delicatessen	department_store
desert/sand	desert/vegetation
desert_road	diner/outdoor
dining_hall	dining_room
discotheque	doorway/outdoor
dorm_room	downtown
dressing_room	driveway
drugstore	elevator/door
elevator_lobby	elevator_shaft
embassy	engine_room
entrance_hall	escalator/indoor
excavation	fabric_store
farm	fastfood_restaurant
field/cultivated	field/wild
field_road	fire_escape
fire_station	fishpond
flea_market/indoor	florist_shop/indoor

food_court	football_field
forest/broadleaf	forest_path
forest_road	formal_garden
fountain	galley
garage/indoor	garage/outdoor
gas_station	gazebo/exterior
general_store/indoor	general_store/outdoor
gift_shop	glacier
golf_course	greenhouse/indoor
greenhouse/outdoor	grotto
gymnasium/indoor	hangar/indoor
hangar/outdoor	harbor
hardware_store	hayfield
heliport	highway
home_office	home_theater
hospital	hospital_room
hot_spring	hotel/outdoor
hotel_room	house
hunting_lodge/outdoor	ice_cream_parlor
ice_floe	ice_shelf
ice_skating_rink/indoor	ice_skating_rink/outdoor
iceberg	igloo
industrial_area	inn/outdoor
islet	jacuzzi/indoor
jail_cell	japanese_garden
jewelry_shop	junkyard
kasbah	kennel/outdoor
kindergarden_classroom	kitchen
lagoon	lake/natural
landfill	landing_deck
laundromat	lawn
lecture_room	legislative_chamber
library/indoor	library/outdoor
lighthouse	living_room
loading_dock	lobby
lock_chamber	locker_room
mansion	manufactured_home
market/indoor	market/outdoor

marsh	martial_arts_gym
mausoleum	medina
mezzanine	moat/water
mosque/outdoor	motel
mountain	mountain_path
mountain_snowy	movie_theater/indoor
museum/indoor	museum/outdoor
music_studio	natural_history_museum
nursery	nursing_home
oast_house	ocean
office	office_building
office_cubicles	oilrig
operating_room	orchard
orchestra_pit	pagoda
palace	pantry
park	parking_garage/indoor
parking_garage/outdoor	parking_lot
pasture	patio
pavilion	pet_shop
pharmacy	phone_booth
physics_laboratory	picnic_area
pier	pizzeria
playground	playroom
plaza	pond
porch	promenade
pub/indoor	racecourse
raceway	raft
railroad_track	rainforest
reception	recreation_room
repair_shop	residential_neighborhood
restaurant	restaurant_kitchen
restaurant_patio	rice_paddy
river	rock_arch
roof_garden	rope_bridge
ruin	runway
sandbox	sauna
schoolhouse	science_museum
server_room	shed

shoe_shop	shopfront
shopping_mall/indoor	shower
ski_resort	ski_slope
sky	skyscraper
slum	snowfield
soccer_field	stable
stadium/baseball	stadium/football
stadium/soccer	stage/indoor
stage/outdoor	staircase
storage_room	street
subway_station/platform	supermarket
sushi_bar	swamp
swimming_hole	swimming_pool/indoor
swimming_pool/outdoor	synagogue/outdoor
television_room	television_studio
temple/asia	throne_room
ticket_booth	topiary_garden
tower	toyshop
train_interior	train_station/platform
tree_farm	tree_house
trench	tundra
underwater/ocean_deep	utility_room
valley	vegetable_garden
veterinarians_office	viaduct
village	vineyard
volcano	volleyball_court/outdoor
waiting_room	water_park
water_tower	waterfall
watering_hole	wave
wet_bar	wheat_field
wind_farm	windmill
yard	youth_hostel
zen_garden	

D Power Consumption and Carbon Emissions

This chapter is purely meant as a rough estimate of the energy consumed during this master's thesis. It is by no means an accurate estimate, as accurate time keeping was not done before the tail end of the project. Regardless we can make some rough estimates. To estimate the power consumption we first need to list out the devices that have used power during the project. We can list up all components that consume a noticable amount of power in the following table:

Device	Average consumption	Hours utilized
Nvidia RTX 3080-Ti	340 W	2500
Nvidia RTX 4090	440 W	15
Nvidia A100[a]	300 W	160

[a] 80 GB version

Given this we can calculate the total power consumed as $340W * 2500Hours + 440W * 15Hours + 300W * 160Hours = 904600Wh = 904.6kWh$. To figure out exactly how much CO_2 was produced by this process, we can find the CO_2 intensity for generating one kWh of energy. Different calculations may yield different statistics here. We opted for using the EU-27 average given by European Environment Agency (2020). The reason for using this data instead of the local Norwegian ones, which is a lot friendlier than the average due to reliance on hydropower (18.92 gCO_2/kWh), is somewhat unrealistic as the Norwegian power grid is heavily integrated with the rest of the European Union. The numbers given are 295.74 gCO_2/kWh, which means the total amount of CO_2 released by this project is roughly $904.6kWh * 295.75g\frac{CO_2}{kWh} = 267535.45gCO_2 = 267.52kgCO_2$.

This number can be hard to grasp, so we can use some comparisons. Using information from Miljødirektoratet[1]. We find that this amount of CO_2 is equivalent to 100.57 liters of diesel. My car consumes an average of 4.8 Liters of diesel per 100km. Meaning I could drive a total of 2,095km and release the same amount of CO_2 as this project generated, or roughly the distance from Gjøvik, Norway to Saint-Étienne, France (2,400km). If we instead considered Norways carbon intensity (gCO_2/kWh), we would end up having about 6.4 liters of diesel. Just enough to get me to from Gjøvik to Oslo, with a few kilometers to spare.

[1] `https://www.miljodirektoratet.no/ansvarsomrader/klima/for-myndigheter/kutte-utslipp-av-klimagasser/klima-og-energiplanlegging/tabeller-for-omregning-fra-energivarer-til-kwh/`

Wild Animal Species Classification from Camera Traps Using Metadata Analysis

Aslak Tøn*, Ali Shariq Imran† and Mohib Ullah‡
Department of Computer Science, Norwegian University of Science and Technology, 2815 Gjøvik, Norway
Email: *aslakto@stud.ntnu.no, †ali.imran@ntnu.no, ‡mohib.ullah@ntnu.no

***Abstract*—Camera trap imaging has emerged as a valuable tool for modern wildlife surveillance, enabling researchers to monitor and study wild animals and their behaviours. However, a significant challenge in camera trap data analysis is the labour-intensive task of species classification from the captured images. This study proposes a novel approach to species classification by leveraging metadata associated with camera trap images. By developing predictive models using metadata alone, we demonstrate that accurate species classification can be achieved without accessing the image data. Our approach reduces the computational burden and offers potential benefits in scenarios where image access is restricted or limited. Our findings highlight the valuable role of metadata in complementing the species classification process and present new opportunities for efficient and scalable wildlife monitoring using camera trap technology.**

***Index Terms*—Metadata, Camera trap imaging, Neural networks, Data fusion, Scene recognition.**

I. Introduction

Human-induced influences like climate change [1], [2], deforestation [3], and trafficked roads [4], [5] have resulted in a dramatic wildlife strain, ushering in an era termed "Anthropocene" [6]. Monitoring such habitats [7], [8] is crucial, as shown by the 2019-20 Australian wildfires [9]. Camera traps offer rich insights [10]–[12], but growing data volumes necessitate robust filtering [13], [14]. Databases like LILA BC and the Snapshot Serengeti (SS) dataset [15] exist, and this paper utilizes a smaller dataset from the Norwegian Institute for Nature Research [16]. Past studies mainly employed image analysis for species identification [13], [14], [17], with few incorporating metadata [18]–[20]. Our study emphasizes metadata's significance, defining explicit metadata as data accompanying the image (like temperature, date, and location) and implicit metadata as indirect information about the image itself (like scene descriptors and attributes), extracted using pre-trained models on the places365 dataset [21]. We advance species classification by using metadata alongside image data, enhancing accuracy in camera trap research. The paper proceeds with: Related work in section II, section III discusses the methodology for data acquisition and how the classification was done, Results and discussion is in section IV, and finally we conclude our findings in section V.

II. Related Works

Although there are numerous papers discussing various aspects of metadata usage, limited attention has been given to its direct application for classification purposes. In this section, we explored related works concerning image classification, explicitly focusing on animals. For example, Norouzzadeh et al. [13] suggest image classification is enhanced by object detection, filtering irrelevant background data without requiring additional resources. They used an existing pre-trained model for object detection, achieving an accuracy of 91.71%, precision of 84.47%, and recall of 84.24%. Animals in each scene were counted via bounding boxes, and the kind of animal in non-empty images was identified. Despite an imbalanced dataset, they achieved high accuracy for the majority of classes and an overall accuracy of 91.37%. The paper also explores active learning methods. Norouzzadeh et al. [14] focuses on animal classification, object counting [22], action recognition [23], and detecting children's presence. Their multi-stage fusion network outperforms a full classifier model, tackling four objectives: animal species classification [24], social interaction [25], animal count [26], and attribute addition [27]. They achieved 96.8% accuracy with VGG [28] network for the first task, top-1 accuracy of 94.9%, and top-5 accuracy of 99.1% for the second. Binned animal count achieved 62.8% accuracy and 83.6% when counting within one bin. Action detection yielded 75.6% accuracy, 84.5% precision, and 80.9% recall. Similarly, Schindler et al. [29] proposes a two-stage fusion network using Mask R-CNN for animal classification and action determination. Temporal data from the video were used for action recognition, with variations of ResNet-18 handling $3 \times T \times H \times W$ frame input. The SlowFast network proposed by [30] underperformed. The authors also present their own accuracy metrics for segmentation, with the best segmentation method achieving 63.8% average precision and 94.1% action detection accuracy.

III. Methodology

A. Acquisition

The acquisition of the NINA Viltkamera dataset metadata is a complex task. All images and their corresponding metadata are publicly available on the Norwegian Institute for Nature Research (NINA) website. However, direct downloading is not feasible due to the extensive number of potential unique URLs. Therefore, we resorted to web scraping to acquire the necessary data. Within the website's interactive map, each camera trap pin held specific metadata. By creating a script, we automated the extraction process of these URLs and their corresponding metadata. Each URL was linked to a JSON object under the "VM" entity on the website. This JSON object

979-8-3503-4218-5/23/$31.00 ©2023 IEEE

contained essential metadata like the filename and a foreign key referencing the species id (NOR: "FK_ArtID"). To link the foreign key with the species name, we utilized the function "vm.arter()". Furthermore, the "vm.lokaliteter()" function was used to map the location ID to its corresponding latitude and longitude. This strategy allowed us to automate the extraction of metadata, which was essential for our study. In total, metadata was collected for 170 thousand camera trap images. These samples were split into 65 original classes. These classes were severely imbalanced, to the point where some classes had one or two samples. To combat this, we employed both class combination and data augmentation. More information on this is discussed in Section III-B. In terms of additional metadata, temperature data was often missing. To fill in these gaps, we used the Norwegian Metrological Institute's Frost API[1]. This API provided temperature data from the nearest weather station to the camera trap. We limited temperature readouts to within a 24-hour window of the image capture time. This still left some missing temperature values (16 thousand samples); these were set to the average temperature of the entire dataset. The date and time were stored as a one-hot encoded vector, dubbed the "datetime" vector. This preserves the cyclical nature present in time data while eliminating any ambiguity that may arise. We first considered a sine curve to represent time, as this would also capture the cyclic nature of time. However, this may have confused, as spring and fall would result in the same values. In the same vein, dawn and dusk would also result in the same values. Latitude and longitude were also included to capture potential geographical variations in animal distribution. It is important to note that the positional data acquired is only approximate, as the locations of the pins are only accurate to within about a kilometre radius. Lastly, implicit metadata was obtained through pre-trained models on the Places365 dataset. This provided us with scene attributes and scene descriptors, which offered extra context for species identification. To prevent computation delays during model training, these attributes were pre-extracted and stored alongside the image metadata.

B. Class Imbalance

As mentioned previously, the 170 thousand data points collected were severely imbalanced. The largest class "Roe Deer" consisted of 53 thousand samples alone, while other classes, like "Lemmings" only had three. The birds were especially prone to low sample size, as each individual species of bird was catalogued. Two methods were used to combat this: Class combination and data augmentation. Class combination combines certain classes, like the different bird species, into one larger super-class. In the case of bird species, we combined them to form the "Bird" superclass. Other classes were similarly combined, "Rodent" became one superclass, as did "Deer". In total, with these combinations, we ended up with 25 classes. Furthermore, to balance out the class representation when running deep learning, we utilized Borderline Synthetic Minority Oversampling Technique (Borderline SMOTE) [31]. Borderline SMOTE generates more valuable sample points than the regular SMOTE algorithm. Borderline SMOTE generates synthetic samples on the boundary region between classes, which gives the network more hard-to-tell samples, which should provide more benefit during training.

[1]https://frost.met.no/index.html

C. Noisy Labels

One issue with this dataset is the lack of validation on the said dataset. Several samples with one given class were, in fact, a different class (see Fig. 1). Unfortunately, due to the sheer number of samples, combined with the lack of relevant expertise from the authors of the paper, reclassifying the animals is infeasible. Luckily, the vast majority of labels are correct, with only around $0.5\% - 1\%$ of labels being wrong.

D. Evaluation metrics

Our study primarily focuses on two significant metrics: Accuracy and the Cohen Kappa Score. **Accuracy** quantifies the fraction of true results (including both true positives TP and true negatives TN) in the total number of samples analyzed. Formally, Accuracy is defined as follows:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Where:

- TP stands for True Positives: the number of samples correctly classified to class y_i.
- TN represents True Negatives: the samples correctly not assigned to class y_i.
- FP is False Positives: the samples wrongly assigned to class y_i but should have been classified to a different class y_j.
- FN denotes False Negatives: the samples that should have been classified to y_i but were classified to y_j.

This metric provides a view of our model's overall performance. Due to the imbalanced nature of our dataset, we opted to use a metric sensitive to prediction accuracy that accounts for class imbalance. Thus, we incorporate the **Cohen Kappa Score**. The Cohen Kappa Score measures the agreement between two raters who classify N items into C mutually exclusive classes. The score calculates the possibility of the agreement occurring by chance (p_e) and the observed agreement (p_o). Initially, the probability of random agreement, p_e, is calculated as:

$$p_e = \frac{1}{N^2} \sum_{k=1}^{C} n_k^{(1)} n_k^{(2)}$$

Here, $n_k^{(i)}$ is the number of times the rater i predicted class k. Next, the observed agreement, p_o, is calculated as:

$$p_o = \frac{\sum_{i=1}^{C} x_{i,i}}{\sum_{i=1}^{C} \sum_{j=1}^{C} x_{i,j}}$$

Fig. 1: Animal misclassifications. All are labelled as “Sheep”

Here, the elements $x_{i,j}$ constitute the observed response matrix M. Finally, the Cohen Kappa Score (κ) is calculated using these probabilities:

$$\kappa = \frac{p_o - p_e}{1 - p_e}$$

This score provides a more robust measure than accuracy as it considers both the class imbalance and the probability of a correct prediction occurring by chance, offering a more nuanced view of our model’s performance.

E. Classification

To properly evaluate what effects metadata would have on classification, we need to perform an exhaustive search of the classes and features available. This involves classifying n classes using m features, where $n \geq 2$ and $m \geq 1$. To run all these combinations, we would have a total of $1,040,186,586$ individual cases to test. This amount of computation is currently unrealistic. Instead, we opted to look at a subset of the classes. The classes we decided to investigate were: ‘Fox’, ‘Deer’, ‘Mustelidae’, ‘Bird’, ‘Lynx’, ‘Cat’, ‘Sheep’, ‘Rodent’, and ‘Wolf’. We also combined temperature and position into one feature. The reasoning is that the single data point of temperature would likely not be a perfect classifier. This left us with nine classes and four features that could be included or excluded. This gives a more manageable 7529 combination that we exhaustively classify. We focused on the quantitative study of all permutations of animals and metadata information. We used a 4-layer fully connected network, with batch normalization and dropout between each layer to combat overfitting. The hidden layers were static, having 64 and 32 neurons, respectively. The input layer had a dynamic number of neurons equal to the number of input features currently selected. Likewise, the output layer was set to the current number of classes to be classified.

F. Data Visualization

Another efficient way of assessing if metadata can be used to classify different species is the use of data visualization tools. Our data consists of 538 data points, meaning we could map the data in a 538-dimensional space and assess what groupings are present in the data. As no currently known technique exists for viewing visual information above three dimensions, four if you include temporal information, we had to rely on dimensionality reduction techniques instead. Dimensionality reduction, in general, aims to preserve the structure of the data as much as possible while reducing the overall information saved for each data point. Our paper utilizes a new approach to dimensionality reduction proposed by [32]. Uniform Manifold Approximation and Projection, or UMAP for short, utilizes topology, higher dimensional manifolds, and graph theory in order to project high dimensional data down to a lower dimension while minimizing the cross entropy between the original projection and the re-projection. The algorithm has been demonstrated to equal or outperform other popular dimensionality reduction techniques such as t-SNE [33], LargeVis [34], and Laplacian eigenmaps [35]. The theory behind UMAP is quite involved, requiring a good understanding of the topic of topology. However, an excellent summary was given by [36]. They break down the process into two major steps and a couple of minor steps in each major step as so:

1 Learn manifold structure
  1.1 Finding nearest neighbours
  1.2 Constructing neighbours graph
    1.2.1 Varying distance
    1.2.2 Local connectivity
    1.2.3 Fuzzy area
    1.2.4 Merging of edges
2 Finding low-dimensional representation
  2.1 Minimum distance
  2.2 Minimizing the cost function

Utilizing UMAP, we can investigate if any patterns emerge on animal clusters. If we find local clusters in the dimensionality-reduced space, we can expect those same patterns to hold in the original 538-dimensional space we cannot investigate.

G. Implementation Details

To create and run the models, we used Python programming language, with PyTorch [37] framework for creating, importing, and training models. The models primarily used categorical cross-entropy [38] as the loss function and the Adam optimizer [39]. The networks were mainly created and trained on a Linux computer using an intel-i9 12900KF, 128 Gigabytes of RAM and an RTX3080-Ti. All weights were randomly initialized, with the optimizer set with an initial

Classes	Features used	Acc	κ
4, 6	Scene attributes	0.948	0.894
6, 12	Position and temperature, Scene attributes	0.982	0.945
4, 6	Places, Position and temperature, Scene attributes	0.967	0.932
6, 12	Datetime, Places, Position and temperature, Scene attributes	0.989	0.964
3, 4, 6	Scene attributes	0.87	0.779
3, 4, 6	Position and temperature, Scene attributes	0.869	0.782
3, 4, 6	Datetime, Places, Scene attributes	0.866	0.775
3, 4, 6	Datetime, Places, Position and temperature, Scene attributes	0.878	0.796
2, 3, 4, 6	Scene attributes	0.696	0.552
3, 4, 6, 12	Position and temperature, Scene attributes	0.731	0.603
3, 4, 6, 12	Datetime, Position and temperature, Scene attributes	0.729	0.614
3, 4, 6, 12	Datetime, Places, Position and temperature, Scene attributes	0.746	0.63

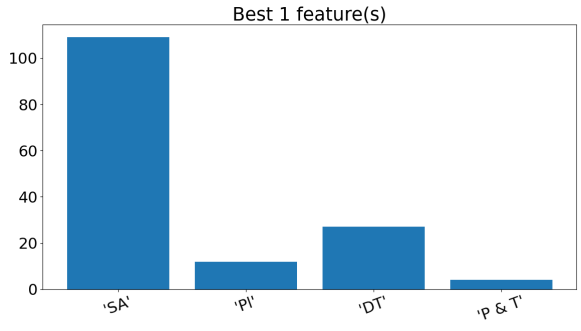
TABLE I: Metadata Predictors Scores

learning rate of $1e-3$. The learning rate was then reduced by an order of magnitude every seven epochs, and a total of 25 epochs ran for each model. The samples were split into mini-batches of 64. For each epoch, the model was validated using 10% of the test samples; if the model performed worse than previous runs, it was reset back to its best-performing iteration. Finally, the model was evaluated using 10% of the data that was left aside before training started.

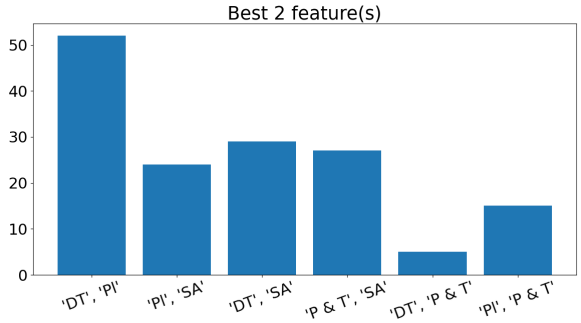
To ensure balanced representation in the training data. Borderline SMOTE [31] was utilized. By having the same number of samples from each class, the network cannot "cheat" by only predicting the majority class to achieve an acceptable result. The validation sets and testing sets were left unaltered.

IV. Results and Discussion

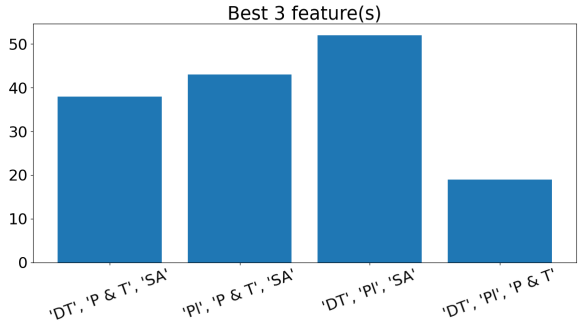
We can see the results for two or three separate classes using one, two, three or all four features. Looking at Table I, we see a reasonably high accuracy for classifying some animal species, despite not having any image data. We've decided to use an ID for each species instead of the said species' name. The corresponding ID to species is 0: 'Fox', 1: 'Deer', 2: 'Weasel', 3: 'Bird', 4: 'Lynx', 5: 'Cat', 6: 'Sheep', 7: 'Squirrel', 8: 'Rabbit', 9: 'Rodent', 10: 'Cattle', 11: 'Boar', 12: 'Wolf', and 13: 'Bear'. We see that the "Scene attributes" information yields the best single feature to include in the prediction. We also see as we increase the number of features included increases, the best performer is still "Scene attributes". However, including extra attributes does yield diminishing returns. The average performance of the different features is less clear-cut. We can quantify this relation better by looking at the "winner" when pitting n predictors against each other to predict between m classes. By finding and counting the best predictor(s) for all combinations of animals, we get Fig. 2. To save space, we used abbreviated versions of the feature names, 'SA' equates to scene attributes, 'Pl' is short for "Places" which are the Scene descriptors, 'DT' is the datetime vector, and 'P & T' is the position and temperature information. We see that "Scene attributes" is the clear best single predictor. However, it is not among the pair of best predictors, being beaten out by the combination of "Datetime" and "Places". Its worth noting that this method of counting the winner does not take into account



(a)



(b)



(c)

Fig. 2: The best n features to use to distinguish a set of m animals

how much better one predictor performed than another. We do not know whether "Scene Features" dominated the competition as the singular feature or if other features were close seconds to the best performance of "Scene Features". However, we can conclude that accuracy, in general, improves when more features are included. Meaning all the metadata contributes something valuable to the prediction of the animal feature. Remember that these predictions of animal classes are purely based on the metadata information, no image of the animal is given to the model, yet it can quite confidently predict between two classes.

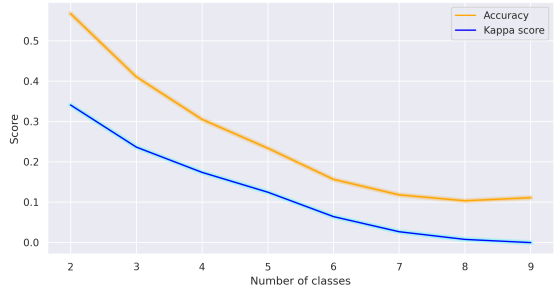
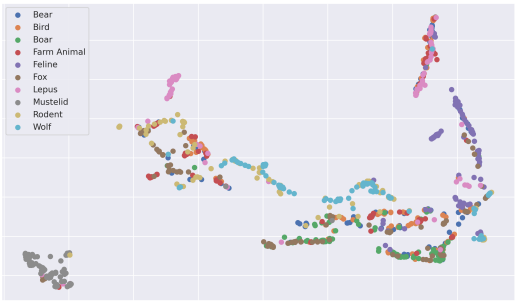
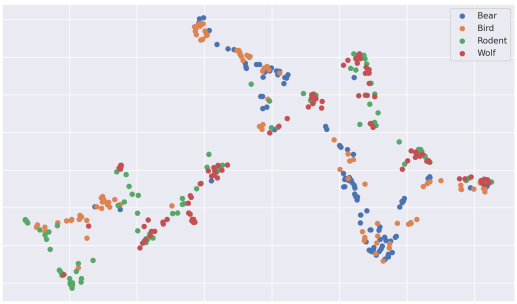


Fig. 3: Prediction score versus the number of classes to distinguish

The prediction score does steadily decrease as more classes are included. Fig. 3 demonstrates this clearly. We postulate this is due to the increased homogeneous actions of the animals. Some animals may be active during the daytime, others during nighttime; some are preferentially spotted in some locations, while others avoid those same locations. When we only have two animals, we can use these facts to separate them. However, once multiple animals act similarly, we can no longer separate them purely using this metadata, and image data are required. This issue of reduced performance when more classes make intuitive sense. It is harder to guess between 5 categories than it is to guess between only two. However, the kappa score should account for this increased performance of randomly guessing the correct class, but it is also declining. Some of the explanations for this can be seen by using UMAP. Fig. 4a shows Mustelidae cleanly separating into its own cluster. This indicates that some higher dimensional line can be drawn that can confidently classify Mustelidae away from other animals. However, once we remove many of the classes, we find that UMAP no longer cleanly separates these classes. This problem can be seen in Fig. 4b. We can summarize that metadata has the ability to help differentiate species from each other without the need for image data to be included. These findings are more valuable when we include image data once again. By designing networks that can incorporate metadata to image feature extraction for networks, we believe we can enhance the classification results over standard network architectures. Metadata should prove even more helpful in cases where there are few classes to choose from or where the existing classes have distinct behavioural patterns that separate them from each other at a metadata level, such as different biomes, locations, or sleep schedules that result in image capture during different hours.



(a) UMAP separating Mustelidae cleanly from other classes



(b) UMAP struggling to separate the remaining classes

Fig. 4: UMAP embedding of metadata features and classes

V. Conclusion

In our study, we have showcased the effectiveness of utilizing explicit and implicit metadata associated with camera trap images for animal prediction. The results obtained highlight the potential of metadata-driven augmentation for deep-learning approaches in the field of animal classification. Building upon these findings, we recommend employing a two-step classification process: First, identifying the appropriate subgroups into which animals can be separated using the available metadata and then utilizing more specific prediction models to assign the final species label to each animal. This coarse-to-fine classification methodology aligns well with the outcomes and implications presented in the paper. Our work holds promise for improving the overall accuracy and efficiency of animal classification in camera trap research.

Acknowledgment

We would like to acknowledge the provision of images by the SCANDCAM project, which is coordinated by the Norwegian Institute for Nature Research and has received funding from the Norwegian Environment Agency and various Norwegian county councils.

REFERENCES

[1] V. Masson-Delmotte, P. Zhai, A. Pirani, S.L. Connors, C. Péan, S. Berger, N. Caud, Y. Chen, L. Goldfarb, M.I. Gomis, M. Huang, K. Leitzell, E. Lonnoy, J.B.R. Matthews, T.K. Maycock, T. Waterfield, O. Yelekçi, R. Yu, and B. Zhou, Eds., *Human Influence on the Climate System*, pp. 423–552, Cambridge University Press, Cambridge, United Kingdom and New York, NY, USA, 2021.

[2] Muhammad Munsif, Hina Afridi, Mohib Ullah, Sultan Daud Khan, Faouzi Alaya Cheikh, and Muhammad Sajjad, "A lightweight convolution neural network for automatic disasters recognition," in *2022 10th European Workshop on Visual Information Processing (EUVIP)*. IEEE, 2022, pp. 1–6.

[3] Kusum Lata, Arvind Kumar Misra, and Jang Bahadur Shukla, "Modeling the effect of deforestation caused by human population pressure on wildlife species," *Nonlinear Analysis: Modelling and Control*, vol. 23, no. 3, pp. 303–320, 2018.

[4] W Richard J Dean, Colleen L Seymour, Grant S Joseph, and Stefan H Foord, "A review of the impacts of roads on wildlife in semi-arid regions," *Diversity*, vol. 11, no. 5, pp. 81, 2019.

[5] Maryam Hassan, Farhan Hussain, Sultan Daud Khan, Mohib Ullah, Mudassar Yamin, and Habib Ullah, "Crowd counting using deep learning based head detection," *Electronic Imaging*, vol. 35, pp. 293–1, 2023.

[6] Simon L Lewis and Mark A Maslin, "Defining the anthropocene," *Nature*, vol. 519, no. 7542, pp. 171–180, 2015.

[7] Joel Berger, Steven L Cain, and Kim Murray Berger, "Connecting the dots: an invariant migration corridor links the holocene to the present," *Biology Letters*, vol. 2, no. 4, pp. 528–531, 2006.

[8] Toby A Patterson, Len Thomas, Chris Wilcox, Otso Ovaskainen, and Jason Matthiopoulos, "State–space models of individual animal movement," *Trends in ecology & evolution*, vol. 23, no. 2, pp. 87–94, 2008.

[9] Isabel T Hyman, Shane T Ahyong, Frank Köhler, Shane F McEvey, GA Milledge, Chris AM Reid, and Jodi JL Rowley, "Impacts of the 2019–2020 bushfires on new south wales biodiversity: a rapid assessment of distribution data for selected invertebrate taxa," *Technical reports of the Australian Museum online*, vol. 32, pp. 1–17, 2020.

[10] Franck Trolliet, Cédric Vermeulen, Marie-Claude Huynen, and Alain Hambuckers, "Use of camera traps for wildlife studies: a review," *Biotechnologie, Agronomie, Société et Environnement*, vol. 18, no. 3, 2014.

[11] Allan F O'Connell, James D Nichols, and K Ullas Karanth, *Camera traps in animal ecology: methods and analyses*, vol. 271, Springer, 2011.

[12] Francesco Rovero, Fridolin Zimmermann, Duccio Berzi, and Paul 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*, 2013.

[13] Mohammad Sadegh Norouzzadeh, Dan Morris, Sara Beery, Neel Joshi, Nebojsa Jojic, and Jeff Clune, "A deep active learning system for species identification and counting in camera trap images," *Methods in ecology and evolution*, vol. 12, no. 1, pp. 150–161, 2021.

[14] Mohammad Sadegh Norouzzadeh, Anh Nguyen, Margaret Kosmala, Alexandra Swanson, Meredith S Palmer, Craig Packer, and Jeff Clune, "Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning," *Proceedings of the National Academy of Sciences*, vol. 115, no. 25, pp. E5716–E5725, 2018.

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

[16] John Odden and Jon E. Swenson, "Scandcam project," https://viltkamera.nina.no/, 2023, Images provided by the SCANDCAM project coordinated by the Norwegian Institute for Nature Research with funding from the Norwegian Environment Agency and multiple Norwegian county councils.

[17] Mohib Ullah, Zolbayar Shagdar, Habib Ullah, and Faouzi Alaya Cheikh, "Semi-supervised principal neighbourhood aggregation model for sar image classification," in *2022 16th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*. IEEE, 2022, pp. 211–217.

[18] John Arevalo, Thamar Solorio, Manuel Montes-y Gómez, and Fabio A. González, "Gated multimodal units for information fusion," *arXiv preprint arXiv:1702.01992*, 2017, Submitted on 7 Feb 2017.

[19] Andre GC Pacheco and Renato A Krohling, "An attention-based mechanism to combine images and metadata in deep learning models applied to skin cancer classification," *IEEE journal of biomedical and health informatics*, vol. 25, no. 9, pp. 3554–3563, 2021.

[20] Weipeng Li, Jiaxin Zhuang, Ruixuan Wang, Jianguo Zhang, and Wei-Shi Zheng, "Fusing metadata and dermoscopy images for skin disease diagnosis," in *2020 IEEE 17th international symposium on biomedical imaging (ISBI)*. IEEE, 2020, pp. 1996–2000.

[21] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba, "Places: A 10 million image database for scene recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[22] Sultan Daud Khan, Ahmed B Altamimi, Mohib Ullah, Habib Ullah, and Faouzi Alaya Cheikh, "Tcm: Temporal consistency model for head detection in complex videos," *Journal of Sensors*, vol. 2020, pp. 1–13, 2020.

[23] Mohib Ullah, Muhammad Mudassar Yamin, Ahmed Mohammed, Sultan Daud Khan, Habib Ullah, and Faouzi Alaya Cheikh, "Attention-based lstm network for action recognition in sports," *Electronic Imaging*, vol. 33, pp. 1–6, 2021.

[24] Tinao Petso, Rodrigo S Jamisola, and Dimane Mpoeleng, "Review on methods used for wildlife species and individual identification," *European Journal of Wildlife Research*, vol. 68, pp. 1–18, 2022.

[25] Habib Ullah, Sultan Daud Khan, Mohib Ullah, and Faouzi Alaya Cheikh, "Social modeling meets virtual reality: An immersive implication," in *Pattern Recognition. ICPR International Workshops and Challenges: Virtual Event, January 10–15, 2021, Proceedings, Part IV*. Springer, 2021, pp. 131–140.

[26] Colin J Torney, David J Lloyd-Jones, Mark Chevallier, David C Moyer, Honori T Maliti, Machoke Mwita, Edward M Kohi, and Grant C Hopcraft, "A comparison of deep learning and citizen science techniques for counting wildlife in aerial survey images," *Methods in Ecology and Evolution*, vol. 10, no. 6, pp. 779–787, 2019.

[27] Milan Kresovic, Thong Nguyen, Mohib Ullah, Hina Afridi, and Faouzi Alaya Cheikh, "Pigpose: A realtime framework for farm animal pose estimation and tracking," in *IFIP International Conference on Artificial Intelligence Applications and Innovations*. Springer, 2022, pp. 204–215.

[28] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[29] Frank Schindler and Volker Steinhage, "Identification of animals and recognition of their actions in wildlife videos using deep learning techniques," *Ecological Informatics*, vol. 61, pp. 101215, 2021.

[30] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He, "Slowfast networks for video recognition," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 6202–6211.

[31] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao, "Borderline-smote: a new over-sampling method in imbalanced data sets learning," in *International conference on intelligent computing*. Springer, 2005, pp. 878–887.

[32] Leland McInnes, John Healy, and James Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," 2020.

[33] Geoffrey E Hinton and Sam Roweis, "Stochastic neighbor embedding," *Advances in neural information processing systems*, vol. 15, 2002.

[34] Jian Tang, Jingzhou Liu, Ming Zhang, and Qiaozhu Mei, "Visualizing large-scale and high-dimensional data," in *Proceedings of the 25th International Conference on World Wide Web*. apr 2016, International World Wide Web Conferences Steering Committee.

[35] Mikhail Belkin and Partha Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[36] Hina Afridi, Mohib Ullah, Øyvind Nordbø, Faouzi Alaya Cheikh, and Anne Guro Larsgard, "Optimized deep-learning-based method for cattle udder traits classification," *Mathematics*, vol. 10, no. 17, pp. 3097, 2022.

[37] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala, "Pytorch: An imperative style, high-performance deep learning library," 2019.

[38] Zhilu Zhang and Mert R. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels," 2018.

[39] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," 2017.

Bibliography

Afridi, H., Ullah, M., Nordbø, Ø., Cheikh, F. A., and Larsgard, A. G. (2022). Optimized deep-learning-based method for cattle udder traits classification. *Mathematics*, 10(17):3097. (cited on page 25)

Anton, V., Hartley, S., Geldenhuis, A., and Wittmer, H. U. (2018). Monitoring the mammalian fauna of urban areas using remote cameras and citizen science. *Journal of Urban Ecology*, 4(1). (cited on page 34)

Arevalo, J., Solorio, T., Montes-y Gómez, M., and González, F. A. (2017). Gated multimodal units for information fusion. *arXiv preprint arXiv:1702.01992*. Submitted on 7 Feb 2017. (cited on pages 36, 38, and 70)

Beery, S., Horn, G. V., and Perona, P. (2018). Recognition in terra incognita. In Ferrari, V., Hebert, M., Sminchisescu, C., and Weiss, Y., editors, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XVI*, volume 11220 of *Lecture Notes in Computer Science*, pages 472–489. Springer. (cited on page 34)

Berger, J., Cain, S. L., and Berger, K. M. (2006). Connecting the dots: an invariant migration corridor links the holocene to the present. *Biology Letters*, 2(4):528–531. (cited on page 1)

Bhatt, C. A. and Kankanhalli, M. S. (2011). Multimedia data mining: state of the art and challenges. *Multimedia Tools and Applications*, 51:35–76. (cited on page 20)

Bi, L., Fulham, M., and Kim, J. (2022). Hyper-fusion network for semi-automatic segmentation of skin lesions. *Medical Image Analysis*, 76:102334. (cited on pages 36 and 38)

Buslaev, A., Parinov, A., Khvedchenya, E., Iglovikov, V. I., and Kalinin, A. A. (2018). Albumentations: fast and flexible image augmentations. *ArXiv e-prints*. (cited on page 48)

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357. (cited on page 20)

Chen, X., Liang, C., Huang, D., Real, E., Wang, K., Liu, Y., Pham, H., Dong, X., Luong, T., Hsieh, C.-J., Lu, Y., and Le, Q. V. (2023). Symbolic discovery of optimization algorithms. (cited on pages 14 and 47)

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46. (cited on page 27)

Cordier, C. P., Smith, D. A. E., Smith, Y. E., and Downs, C. T. (2022). Camera trap research in africa: a systematic review to show trends in wildlife monitoring and its value as a research tool. *Global Ecology and Conservation*, page e02326. (cited on page 1)

de Lima, D. C., Saqui, D., Mpinda, S. A. T., and Saito, J. H. (2022). Pix2pix network to estimate agricultural near infrared images from rgb data. *Canadian Journal of Remote Sensing*, 48(2):299–315. (cited on pages 37 and 38)

Dean, W. R. J., Seymour, C. L., Joseph, G. S., and Foord, S. H. (2019). A review of the impacts of roads on wildlife in semi-arid regions. *Diversity*, 11(5):81. (cited on page 1)

DeVries, T. and Taylor, G. W. (2017). Improved regularization of convolutional neural networks with cutout. (cited on page 23)

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7). (cited on page 13)

European Environment Agency (2020). Co2 intensity of electricity generation. European Environment Agency (EEA). Prod-ID: DAT-232-en. (cited on page 93)

Falzon, G., Lawson, C., Cheung, K.-W., Vernes, K., Ballard, G. A., Fleming, P. J., Glen, A. S., Milne, H., Mather-Zardain, A., and Meek, P. D. (2019). Classifyme: a field-scouting software for the identification of wildlife in camera trap images. *Animals*, 10(1):58. (cited on pages 31 and 37)

Feichtenhofer, C., Fan, H., Malik, J., and He, K. (2019). Slowfast networks for video recognition. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6202–6211. (cited on pages 35 and 38)

Flock, W. L. and Green, J. L. (1974). The detection and identification of birds in flight, using coherent and noncoherent radars. *Proceedings of the IEEE*, 62(6):745–753. (cited on page 1)

Garcia-Sanchez, A.-J., Garcia-Sanchez, F., Losilla, F., Kulakowski, P., Garcia-Haro, J., Rodríguez, A., López-Bao, J.-V., and Palomares, F. (2010). Wireless sensor network deployment for monitoring wildlife passages. *Sensors*, 10(8):7236–7262. (cited on page 1)

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press. (cited on pages 6, 7, 8, and 11)

Habib, B., Shrotriya, S., Sivakumar, K., Sinha, P. R., and Mathur, V. B. (2014). Three decades of wildlife radio telemetry in india: a review. *Animal Biotelemetry*, 2:1–10. (cited on page 1)

Han, H., Wang, W.-Y., and Mao, B.-H. (2005). Borderline-smote: a new over-sampling method in imbalanced data sets learning. In *International conference on intelligent computing*, pages 878–887. Springer. (cited on page 21)

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778. (cited on pages 18 and 109)

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. (cited on page 17)

Hyman, I. T., Ahyong, S. T., Köhler, F., McEvey, S. F., Milledge, G., Reid, C. A., and Rowley, J. J. (2020). Impacts of the 2019–2020 bushfires on new south wales biodiversity: a rapid assessment of distribution data for selected invertebrate taxa. *Technical reports of the Australian Museum online*, 32:1–17. (cited on page 1)

Idaho Department of Fish and Game (2021). Idaho department of fish and game camera traps. `https://lilablobssc.blob.core.windows.net/idaho-camera-traps/public/`. (cited on page 34)

Island Conservation, D. W. (2020). Island conservation camera traps. `https://lilablobssc.blob.core.windows.net/islandconservationcameratraps/public`. (cited on page 34)

Kingma, D. P. and Ba, J. (2017). Adam: A method for stochastic optimization. (cited on pages 13 and 47)

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C., Bottou, L., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc. (cited on page 16)

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90. (cited on page 6)

Lata, K., Misra, A. K., and Shukla, J. B. (2018). Modeling the effect of deforestation caused by human population pressure on wildlife species. *Nonlinear Analysis: Modelling and Control*, 23(3):303–320. (cited on page 1)

Leach, P., Mealling, M., and Salz, R. (2005). A universally unique identifier (uuid) urn namespace. RFC 4122, RFC Editor. Also available online at `https://www.rfc-editor.org/rfc/rfc4122.txt`. (cited on page 40)

Lewis, S. L. and Maslin, M. A. (2015). Defining the anthropocene. *Nature*, 519(7542):171–180. (cited on page 1)

Li, W., Zhuang, J., Wang, R., Zhang, J., and Zheng, W.-S. (2020). Fusing metadata and dermoscopy images for skin disease diagnosis. In *2020 IEEE 17th international symposium on biomedical imaging (ISBI)*, pages 1996–2000. IEEE. (cited on pages 36, 38, 51, and 70)

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2018). Focal loss for dense object detection. (cited on pages 9 and 47)

Liu, S. (2018). *Deep learning based multi-modal image analysis for enhanced situation awareness and environmental perception.* PhD thesis, University of British Columbia. (cited on pages 37, 38, 51, and 70)

Masson-Delmotte, V., Zhai, P., Pirani, A., Connors, S., Péan, C., Berger, S., Caud, N., Chen, Y., Goldfarb, L., Gomis, M., Huang, M., Leitzell, K., Lonnoy, E., Matthews, J., Maycock, T., Waterfield, T., Yelekçi, O., Yu, R., and Zhou, B., editors (2021). *Human Influence on the Climate System*, pages 423–552. Cambridge University Press, Cambridge, United Kingdom and New York, NY, USA. (cited on page 1)

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133. (cited on pages 6 and 8)

McInnes, L., Healy, J., and Melville, J. (2020). Umap: Uniform manifold approximation and projection for dimension reduction. (cited on page 23)

Meek, P., Ballard, G., Claridge, A., Kays, R., Moseby, K., O'brien, T., O'connell, A., Sanderson, J., Swann, D., Tobler, M., et al. (2014). Recommended guiding principles for reporting on camera trapping research. *Biodiversity and conservation*, 23(9):2321–2343. (cited on pages 31 and 37)

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814. (cited on page 16)

Njamasi, Y. R., Ndibalema, V. G., and Kioko, J. (2022). The influence of human activities on wildlife in kwakuchinja migratory corridor, tarangire/manyara ecosystem, northern tanzania. *International Journal of Tropical Drylands*, 6(1). (cited on page 1)

Norouzzadeh, M. S., Morris, D., Beery, S., Joshi, N., Jojic, N., and Clune, J. (2021). A deep active learning system for species identification and counting in camera trap images. *Methods in ecology and evolution*, 12(1):150–161. (cited on pages 33 and 38)

Norouzzadeh, M. S., Nguyen, A., Kosmala, M., Swanson, A., Palmer, M. S., Packer, C., and Clune, J. (2018). Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning. *Proceedings of the National Academy of Sciences*, 115(25):E5716–E5725. (cited on pages 35 and 38)

Odden, J. and Swenson, J. E. (2023). Scandcam project. `https://viltkamera.nina.no/`. Images provided by the SCANDCAM project coordinated by the Norwegian Institute for Nature Research with funding from the Norwegian Environment Agency and multiple Norwegian county councils. (cited on page 5)

Pacheco, A. G. and Krohling, R. A. (2021). An attention-based mechanism to combine images and metadata in deep learning models applied to skin cancer classification. *IEEE journal of biomedical and health informatics*, 25(9):3554–3563. (cited on pages 36 and 38)

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. (cited on pages 47 and 48)

Patterson, T. A., Thomas, L., Wilcox, C., Ovaskainen, O., and Matthiopoulos, J. (2008). State–space models of individual animal movement. *Trends in ecology & evolution*, 23(2):87–94. (cited on page 1)

Pievani, T. (2014). The sixth mass extinction: Anthropocene and the human impact on biodiversity. *Rendiconti Lincei*, 25:85–93. (cited on page 1)

Pörtner, H.-O., Roberts, D., Tignor, M., Poloczanska, E., Mintenbeck, K., Alegría, A., Craig, M., Langsdorf, S., Löschke, S., Möller, V., Okem, A., and Rama, B., editors (2022). *Cross-Chapter Paper 1: Biodiversity Hotspots*, pages 2123–2161. Cambridge University Press, Cambridge, UK and New York, NY, USA. (cited on page 1)

Recio, M. R., Mathieu, R., Denys, P., Sirguey, P., and Seddon, P. J. (2011). Lightweight gps-tags, one giant leap for wildlife tracking? an assessment approach. *PloS one*, 6(12):e28225. (cited on page 1)

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386. (cited on pages 6, 7, and 8)

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536. (cited on page 11)

Schindler, F. and Steinhage, V. (2021). Identification of animals and recognition of their actions in wildlife videos using deep learning techniques. *Ecological Informatics*, 61:101215. (cited on pages 35 and 38)

Shorten, C. and Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48. (cited on page 20)

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*. (cited on page 35)

Snapshot Camdeboo (2019). Snapshot camdeboo camera trap images. `https://lilablobssc.blob.core.windows.net/snapshot-safari/CDB/CDB_public`. (cited on page 34)

Snapshot Enonkishu (2019). Snapshot enonkishu camera trap images. `https://lilablobssc.blob.core.windows.net/snapshot-safari/ENO/ENO_public`. (cited on page 34)

Snapshot Karoo (2019). Snapshot karoo camera trap images. https://lilablobssc.blob.core.windows.net/snapshot-safari/KAR/KAR_public. (cited on page 34)

Snapshot Kgalagadi (2019). Snapshot kgalagadi camera trap images. https://lilablobssc.blob.core.windows.net/snapshot-safari/KGA/KGA_public. (cited on page 34)

Snapshot Kruger (2019). Snapshot kruger camera trap images. https://lilablobssc.blob.core.windows.net/snapshot-safari/KRU/KRU_public. (cited on page 34)

Snapshot Mountain Zebra (2019). Snapshot mountain zebra camera trap images. https://lilablobssc.blob.core.windows.net/snapshot-safari/MTZ/MTZ_public. (cited on page 34)

Swanson, A., Kosmala, M., Lintott, C., Simpson, R., Smith, A., and Packer, C. (2015). Data from: Snapshot serengeti, high-frequency annotated camera trap images of 40 mammalian species in an african savanna. (cited on pages 2, 34, 39, and 57)

SWG (2021). Northern and central annamites camera traps 2.0: Iucn ssc asian wild cattle specialist group's saola working group. https://lilablobssc.blob.core.windows.net/swg-camera-traps. (cited on page 34)

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2014). Going deeper with convolutions. (cited on page 17)

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2015). Rethinking the inception architecture for computer vision. (cited on page 17)

Tabak, M. A., Norouzzadeh, M. S., Wolfson, D. W., Sweeney, S. J., VerCauteren, K. C., Snow, N. P., Halseth, J. M., Di Salvo, P. A., Lewis, J. S., White, M. D., et al. (2019). Machine learning to classify animal species in camera trap images: Applications in ecology. *Methods in Ecology and Evolution*, 10(4):585–590. (cited on page 34)

Tan, M. and Le, Q. V. (2020). Efficientnet: Rethinking model scaling for convolutional neural networks. (cited on page 18)

The Nature Conservancy (2021). Channel islands camera traps 1.0. https://lilablobssc.blob.core.windows.net/channel-islands-camera-traps/images. (cited on page 34)

Vélez, J., McShea, W., Shamon, H., Castiblanco-Camacho, P. J., Tabak, M. A., Chalmers, C., Fergus, P., and Fieberg, J. (2023). An evaluation of platforms for processing camera-trap data using artificial intelligence. *Methods in Ecology and Evolution*, 14(2):459–477. (cited on page 34)

Wildlife Conservation Society (2019). Wcs camera traps. `https://lila.science/datasets/wcs-camera-traps`. (cited on page 34)

Woo, S., Park, J., Lee, J.-Y., and Kweon, I. S. (2018). Cbam: Convolutional block attention module. (cited on pages 19 and 109)

Yousif, H., Kays, R., and He, Z. (2019). Dynamic programming selection of object proposals for sequence-level animal species classification in the wild. *IEEE Transactions on Circuits and Systems for Video Technology*. (cited on page 34)

Zeiler, M. D. (2012). Adadelta: An adaptive learning rate method. (cited on page 47)

Zhang, Z., He, Z., Cao, G., and Cao, W. (2016). Animal detection from highly cluttered natural scenes using spatiotemporal object region proposals and patch verification. *IEEE Transactions on Multimedia*, 18(10):2079–2092. (cited on page 34)

Zhang, Z. and Sabuncu, M. R. (2018). Generalized cross entropy loss for training deep neural networks with noisy labels. (cited on page 47)

Zhou, B., Lapedriza, A., Khosla, A., Oliva, A., and Torralba, A. (2017). Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. (cited on pages 46 and 78)

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., and He, Q. (2020). A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76. (cited on page 16)

List of Figures

List of Tables